

AD-A066 807 BDM CORP ALBUQUERQUE N MEX
EXEMPT PROGRAMMERS MANUAL. (U)
JAN 79

F/G 20/14

UNCLASSIFIED

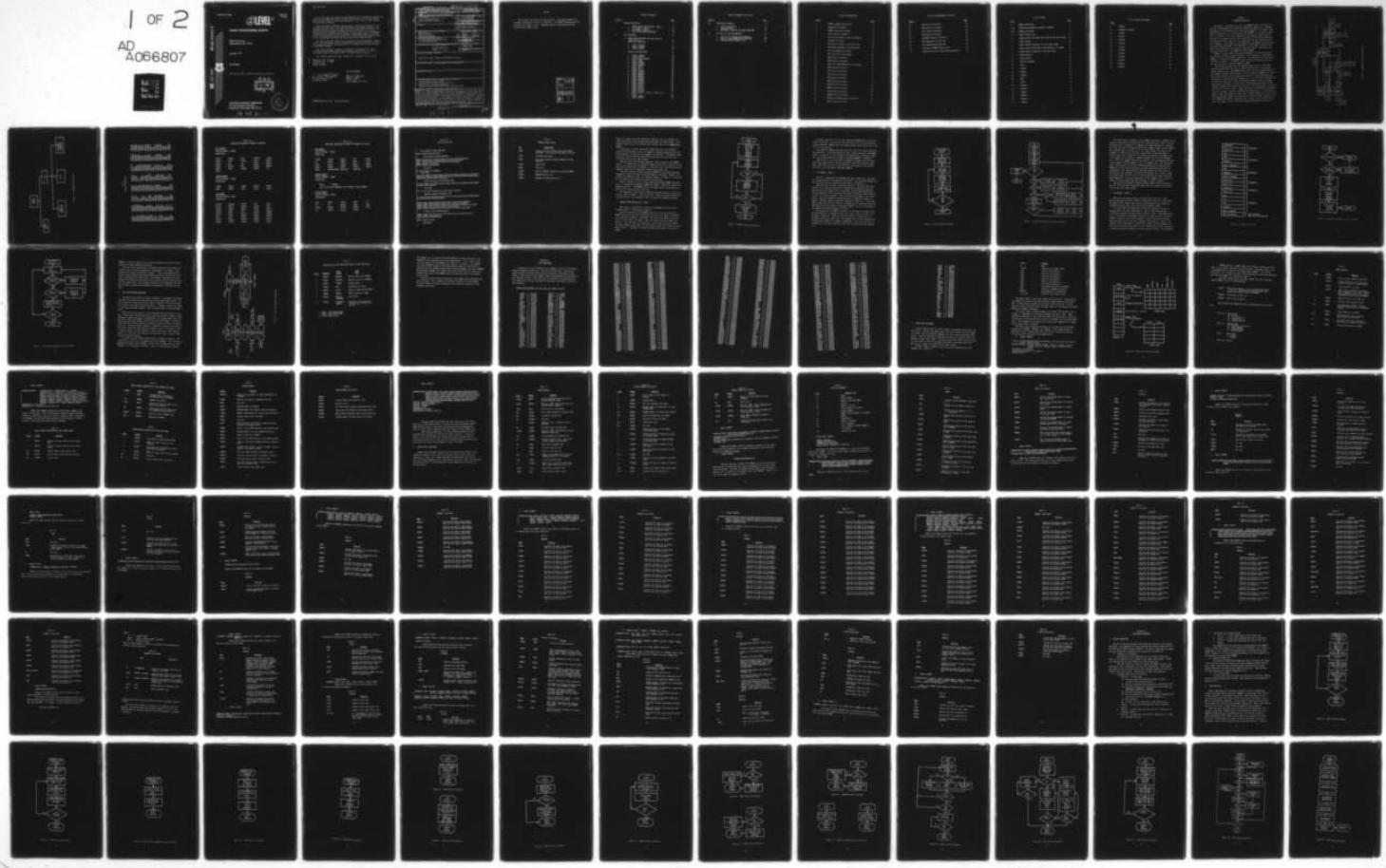
BDM/A-77-098-TR-R2

AFWL-TR-77-206

F29601-76-C-0122

NL

1 OF 2
AD
A066807



AFWL-TR-77-206

AFWL-TR-
77-206

② LEVEL ^{III}

AD8200261

AD A0 66807

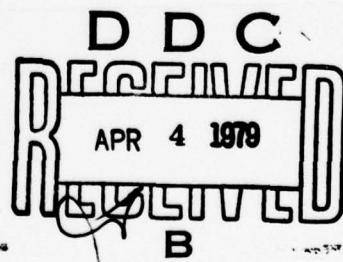
EXEMPT PROGRAMMERS MANUAL

BDM Corporation
Albuquerque, NM 87106

January 1979

Final Report

Approved for public release; distribution unlimited.



AIR FORCE WEAPONS LABORATORY
Air Force Systems Command
Kirtland Air Force Base, NM 87117

79 03 22 068

This final report was prepared by the BDM Corporation, Albuquerque, New Mexico, under Contract F29601-76-C-0122, Job Order 37630113 with the Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico. Mr R. M. Pelzl (ELT) was the Project Officer-in-Charge.

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been authored by a contractor of the US Government. Accordingly, the US Government retains a nonexclusive royalty-free license to publish or reproduce the material contained herein, or allow others to do so, for the US Government purposes.

This report has been reviewed by the Office of Information (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Robert M. Pelzl

ROBERT M. PELZL
Project Officer

FOR THE COMMANDER

J. Philip Castillo
J. PHILIP CASTILLO, PhD
Technology Branch

Donald A. Dowler
DONALD A. DOWLER
Colonel, USAF
Electromagnetics Division

DO NOT RETURN THIS COPY. RETAIN OR DESTROY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(18) AFWL, SBI

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
AFWL-TR-77-206, HD-E209 2611		
4. TITLE (and subtitle)	5. TYPE OF REPORT & PERIOD COVERED	
(6) EXEMPT PROGRAMMERS MANUAL	Final Report	
7. AUTHOR(s)	6. PERFORMING ORG. REPORT NUMBER	
	BDM/A-77-098-TR-R2	
8. CONTRACT OR GRANT NUMBER(s)	7. CONTRACT OR GRANT NUMBER(s)	
	(15) F29601-76-C-0122 new	
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
BDM Corporation 2600 Yale Blvd SE Albuquerque NM 87106	(16) 64711F 37630113 (17) 01	
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE	
Air Force Weapons Laboratory (ELT) Kirtland AFB, NM 87117	(11) January 1979	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	13. NUMBER OF PAGES	
(12) 107P.	98	
16. DISTRIBUTION STATEMENT (of this Report)	15. SECURITY CLASS. (of this report)	
Approved for public release; distribution unlimited.	UNCLASSIFIED	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	18a. DECLASSIFICATION/ DOWNGRADING SCHEDULE	
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Aircraft susceptibility to EMP System analysis of EMP effects on aircraft EMP coupling to aircraft subsystems Computer program modeling of aircraft response to EMP		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
EXEMPT is a computer program which functions as an executive routine that interfaces with computer program models describing electromagnetic pulse (EMP) coupling to aircraft subsystems. As it now exists, EXEMPT is a modification of an earlier version which was difficult to use and which could be used only with B-1 aircraft subsystem models. It was necessary to modify that version to produce a code which would be more user oriented and which could interface with computer program models of EMP coupling to other aircraft subsystems. These modifications have been accomplished, and this report documents the result.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

391884 SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

79 03 22 068

JP

PREFACE

This document was prepared by BDM personnel. The program manager and principal investigator was Mr. R. J. Balestri. The EXEMPT Program Manual was written by Mr. Roger L. Tyler. AFWL/ELA technical coordination has been provided by Mr. Robert Pelzl.

ACCESSION for		
NTIS	White Section <input checked="" type="checkbox"/>	
DDC	Buff Section <input type="checkbox"/>	
UNANNOUNCED <input type="checkbox"/>		
JUSTIFICATION _____		
BY _____		
DISTRIBUTION/AVAILABILITY CODES		
Dist. AVAIL and/or SPECIAL		
A		

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
I PROGRAM DESCRIPTION	1
1. EXEMPT DRIVER DESCRIPTION - LEVEL I	9
2. LIP DRIVER - LEVEL II	11
3. CLP DRIVER - LEVEL II	14
4. DATA FLOW BETWEEN PROCESSORS	18
II DATA STRUCTURES	22
1. COMMON BLOCK GLOSSARY (Variable Name and Common Location)	22
2. MAJOR DATA STRUCTURES	26
a. Common /SYMDAT/	27
b. Common /PARTBL/	31
c. Common /SCNTBL/	35
3. SUPPORT DATA STRUCTURES	35
a. Common /ANGDAT/	38
b. Common /ARGTBL/	38
c. Blank Common /BLNKCM/	39
d. Common /CHAR/	39
e. Common /DEFLST/	41
f. Common /DSKNAM/	43
g. Common /ERMODE/	43
h. Common /ERR/	45
i. Common /FITS/	45
j. Common /HOLSTR/	46
k. Common /KEYWRD/	47
l. Common /KWNAM1/	48
m. Common /KWNAM2/	50
n. Common /KWNAM3/	52
o. Common /KWNAM4/	54
p. Common /KWNAM5/	57
q. Common /KWTASK/	59
r. Common /LOCDAT/	61
s. Common /MSTRG/	61
t. Common /PLTFMT/	62
u. Common /PLTLAB/	63
v. Common /PLTLGV/	63
w. Common /PLTCM1/, PLTCM2/, PLTCM3/, and PLTCM4	65
x. Common /TMPFIL/	67
y. Common /XQTTBL/	68

TABLE OF CONTENTS (Concluded)

<u>Section</u>		<u>Page</u>
III	FUNCTIONAL FLOWCHARTS	70
	1. LOGICAL OPERATIONS	70
	2. SLAM FUNCTIONS	71
	3. ARITHMETIC, FORTRAN, OR EXEMPT FUNCTIONS	87
IV	CDC 6600 and 7600 REQUIREMENTS	92
	1. 6600 TO 7600 CONVERSION REQUIREMENTS	92
	2. 6600 AND 7600 SEGMENTATION DIRECTIVES	93
	3. BUILDING EXEMPT ABSOLUTE FILE	93

LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
1	EXEMPT Language Architecture	2
2	Segmentation Structure	3
3	EXEMPT Functional Flowchart	10
4	LIP Functional Flowchart	12
5	Functional Flowchart of the LIP Scanner	13
6	Argument List Table	15
7	Functional Flowchart of the LIP Parser	16
8	Functional Flowchart of the CLP Driver	17
9	Data Flow Between Processors	19
10	Major Data Structure Linkage	28
11	READ Function Flowchart	72
12	WRITE Function Flowchart	73
13	DROP, EOF, AND REWIND Function Flowchart	74
14	SKIPB Function Flowchart	75
15	SKIPP Function Flowchart	76
16	FILE Function Flowchart	77
17	SAVE Function Flowchart	77
18	PURGE Function Flowchart	78
19	DELETE Function Flowchart	79
20	READF Function Flowchart	80
21	TITLE Function Flowchart	80
22	RENAME Function Flowchart	81
23	GETRAN and SAVRAN Function Flowcharts	81
24	READC Function Flowchart	82

LIST OF ILLUSTRATIONS (Concluded)

<u>Figure</u>		<u>Page</u>
25	LIST Function Flowchart	83
26	PUNCH Function Flowchart	84
27	PLOT Function Flowchart	85
28	SLAM Function Flowchart	86
29	ARITHMETIC Function Flowchart	88
30	6600 Segmentation Directives	94
31	7600 Segmentation Directives	95
32	Creating an EXEMPT Absolute File	96
33	Absolute File With a User-Defined Subroutine	97

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1	EXEMPT SUBROUTINES	4
2(a)	SUBROUTINE BREAKDOWN OF EXEMPT BY SEGMENT	5
2(b)	SEGLOAD DIRECTIVES	7
3	EXEMPT LOGICAL FILES	8
4	ASSOCIATION OF THE CODED FIELD TABLE TO THE FIELD TABLE	20
5	SYMDAT ENTRIES	30
6	PARTBL ENTRIES PERTAINING TO THE LITERAL TABLE	31
7	PARTBL ENTRIES PERTAINING TO THE ARGUMENT LIST TABLE	32
8	PARTBL ENTRIES PERTAINING TO THE LOOP TABLE	32
9	PARTBL ENTRIES	33
10	SCNTBL ENTRIES	36
11	TYPES OF ARGUMENTS	39
12	/CHAR/	40
13	/DEFLST/	42
14	/DSKNAM	43
15	/ERMODE/	44
16	/ERR/	45
17	/FITS/	46
18	/HOLSTR/	47
19	/KEYWRD/	47
20	/KWNAM1/	48
21	/KWNAM2/	50
22	/KWNAM3/	52
23	/KWNAM4/	54

LIST OF TABLES (CONCLUDED)

<u>Table</u>		<u>Page</u>
24	/KWNAM5/	57
25	ARGUMENT LIST TABLE	60
26	/LOCDAT/	61
27	/MSTRG/	62
28	/PLTFMT/	62
29	/PLTLAB/	63
30	/PLTLGV/	63
31	/PLTCM1/	65
32	/PLTCM2/	66
33	/PLTCM3/	66
34	/PLTCM4/	67
35	/TMPFIL/	68
36	/XQTTBL/	68

SECTION I PROGRAM DESCRIPTION

This manual is intended as an aid to the EXEMPT program listing for an individual knowledgeable with the use of EXEMPT and competent as a FORTRAN programmer on the CDC 6000 and CDC 7000 series computers. This manual, when used in conjunction with a listing, will allow EXEMPT to be supported, modified, or extended. Due to the complexity of the program, these functions should not be attempted without first obtaining a detailed understanding of the program structure, data flow, and functional flow when executing particular commands. The program is well documented with comments indicating the operations being performed. The remainder of this document identifies the data variable and their functions for each major module. Functional flowcharts are provided for each EXEMPT and SLAM command.

EXEMPT is a high order user-oriented language to be used as an aid to the analyst in solving transfer functions in the frequency or time domain. EXEMPT is a modularized task-oriented code. The architecture of the code is presented in Figure 1. The EXEMPT driver initializes the processing and is the main interface to the host computer system.

The processing begins with the Language Input Processor (LIP) which reads and decodes each card until an END or RUN directive is encountered. Control then returns to the EXEMPT driver which will pass program control to the Command Language Processor (CLP). The CLP reads the argument list generated by the LIP and calls the appropriate module to perform the operation. Two passes are made through the argument list. The first pass determines the correctness of the user's input by checking the syntax of each statement. Each task is performed to a point short of manipulating data and/or establishing peripheral files needed for execution. The second pass is the actual execution of the user's input. Symbol attributes are established, peripheral files are assigned, and data are manipulated during pass 2. At this point, a symbol's data are assigned storage; single element symbols are stored in core while multielement symbols are stored on the current random access disk file. Since EXEMPT may encounter large data sets, random access cuts down on the core size needed. Data are retrieved from the random access system and

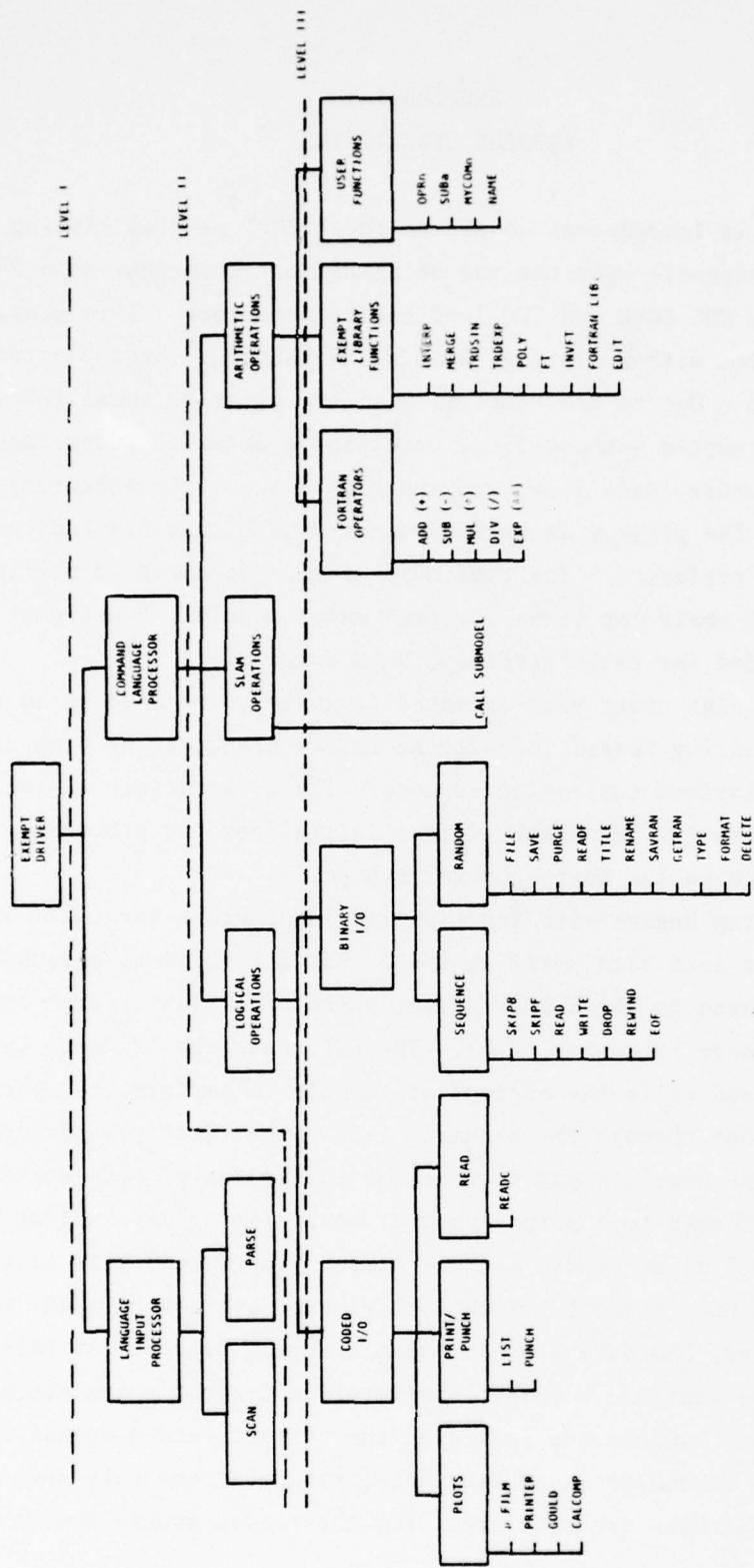


Figure 1. EXEMPT Program Architecture

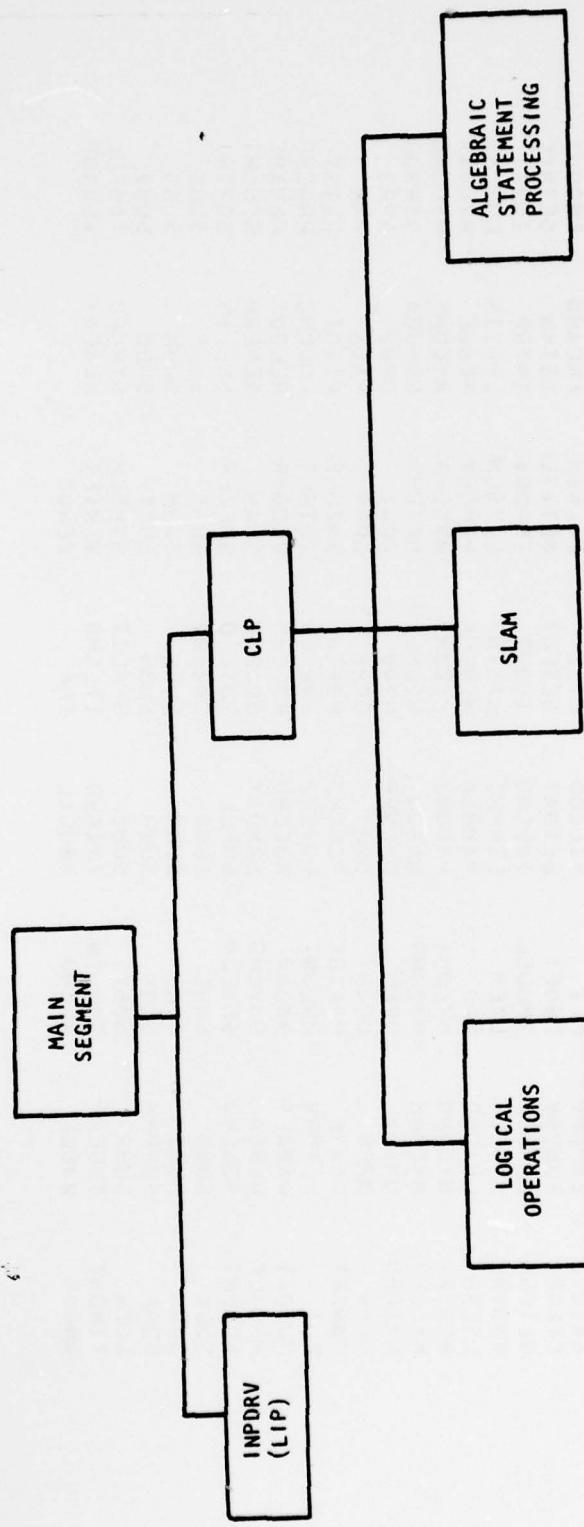


Figure 2. Segmentation Structure

Table 1
EXEMPT SUBROUTINES

EXEMPT	BLKDAT	AITKEN	ARITHOP
AVBAY	AVGSKT	AXNUM	BLKZ
CKFMT	CLOSEPP	CNGSUB	CNPT
CONVRT	COUPLER	CTOUT	COAX
DSKCLS	DSKPN	EDDRV	DBCA1
EXINTO	EXPDRV	FFT	EL
FTFDRV	FUNDRV	FWDFT	ENDPTS
GETPRM	HFANT	IMODEL	FILCOO
KWDVAL	LABEL	LAEM	FILE
LSERR	LSTCMD	MAG	FILNAM
MRGDRV	MYCOM0	MYCOM1	FNDARG
MYCOM7	MYCOM8	MYCOM2	GETKW
NXTOPR	OMEGA	NAMUN	GETTITL
OPR4	OPR5	OPENPP	INPDRV
PARPRT	PARSE	OPR6	INOUT
PLT	PLTDRV	OPR7	LISTO
PUTDAT	RADALT	PARTSK	LITSCHI
RSCALE	RVBEA	PNLJNT	MEMORY
SCNPRT	SEQCMD	RADAR	MINMAX
SUBA	SUBB	RDCCMD	MYCOM5
SUBH	SUBI	SAVCMD	NOSUBA
SUBP	SUBP	SAVAT	NXTCHR
SUBW	SUBX	SFIELD	OPR1
TIMDAT	TRDEXP	SHELL	OPR2
WROOT	W4ELL	SUBD	OPR3
		SUBE	PACK
		SUBL	PAR
		SUBM	PLIST
		SUBS	PITOT
		SUBZ	PUNCHC
		SYMLIT	PRGCMD
		TRDSIN	RELOUT
		WWGND	RENAME
			SCNLMT
			SORTHI
			SNGLIN
			SUBF
			SUBG
			SUBN
			SUBO
			SUBV
			SYMPD
			TABFUL
			WEABAY
			ZEROX

Table 2(a)
SUBROUTINE BREAKDOWN OF EXEMPT BY SEGMENT

MAIN SEGMENT

DRIVING PROGRAM - EXEMPT

SUBROUTINES:

ALTFIL	BLKDAT	BLKZ	CLOSEPP	CLOSRM
CNGSUB	CNTPT	FILCOD	GETKW	GETMYC
KWDVAL	MAG	PAR	PLTDAT	PRESCN
RECOVR	RSCALE	SCAN	SCNERR	SCNLMT
SCNPRT	SHELL	SYMUPD	TRIKWD	YNC
ZEROX				

INPDRV SEGMENT

DRIVING PROGRAM - INPDRV

SUBROUTINES:

DIRMAN	FNDARG	LIPPRT	LITSCH	PARPRT
PARSE	PARTSK	PLIST	SYMLIT	SYMSCH

CLP SEGMENT

DRIVING PROGRAM - EXECUT

SUBROUTINES:

ASINFIT	CHNGLFN	CONVRT	CPUN	CTOUT
DELCMD	DSKCLS	DSKOPN	EXINTO	FILE
FILNAM	FRQOUT	GETDAT	GETFET	GETITL
GETPRM	INOUT	LIST	LSTCMD	MEMCHK
MEMORY	MERGE	MRGDRV	NAMUN	NXTOPR
OPEN	PACK	PCHCMD	PRGCMD	PUNCHC
PUTDAT	RDCCMD	RENAME	RDFCMD	RELOUT
RENAME	SAVCMD	SEQCMD	TTLCMD	VERIFY

Table 2(a)
SUBROUTINE BREAKDOWN OF EXEMPT BY SEGMENT (Concluded)

DMP SEGMENT

DRIVING PROGRAM - EXPDRV

SUBROUTINES:

AITKEN	ARITHOP	BADIDX	EDIT	EDTDRV
FFT	FTFDRV	FUNDRV	FWDFT	INTRP
INVFT	LODLST	NTRPDR	OMEGA	PHZSHF
POWER2	SHFZER	TABFUL	TRDEXP	TRDSIN
XBJCAL	MYCOM0-MYCOM9	OPR0-OPR9	SUBA-SUBZ	

SUBDRV SEGMENT

DRIVING PROGRAM - SUBDRV

SUBROUTINES:

SAVDAT SUBPRM

ALSO, ALL B1/SLAM SUBMODELS ARE INCLUDED IN THIS SEGMENT.

PLTFIL SEGMENT

DRIVING PROGRAM - PLTDRV

SUBROUTINES:

AXIS	AXNUM	CHRCNT	CKFMT	ELI
EL2	EL3	ENDPTS	LABEL	LAEM
LOGLIM	MINMAX	NXTCHR	OPENPP	PDATA
PLT	PRTPLT	SCALE2	TIMDAT	

Table 2(b)
SEGLOAD DIRECTIVES

- MAIN SEGMENT EXEMPT ROUTINES
- EXEMPT INCLUDE BLKDAT,PLTDAT
- MAIN SEGMENT SYSTEM ROUTINES
- EXEMPT INCLUDE GET.WA,OPEN.WA,CLSF.WA,PUT.WA,PTWR.SQ,GTWR.SQ
- EXEMPT INCLUDE SKBL.SQ,SKSB.SQ,CHEK.RM,OPES.SQ,SKSF.SQ
- EXEMPT INCLUDE SKFF.SQ,SKFP.SQ,WTHK.SQ,W.SQ,MEM6RM
- EXEMPT INCLUDE DT.SQ,R.SQ
- EXEMPT INCLUDE PLOTS,GET,RM
- RUN TREE
- RUN TREE EXEMPT-(CLP,INPDRV)
- SLAM TREE
- SLAM TREE SUBDRV-(AS1918,AVBAY,COAX,COCKPT,CONDUIT,ENGINE,HFANT,INDCOU,M,ARBEA,PITOT,RADALT,RADAR,RVBEA,SHIELD,SNGLIN,WEABAY,WEBAOP,WROOT,WWELL,WWGND,SFIELD,AVGSKT,PNLJNT,COUPLER)
- DIRECT MANIPULATION PROCESSOR TREE
- DMP TREE EXPDRV-(ARITHOP,EDTDRV,FTFDRV,FUNDRV,LODLST,MRGDRV,NTRPDR,OMEGA,PHZSHF,TRDEXP,TRDSIN)
- PLOT TREE
- PLTFIL TREE PLTDRV-(OPENPP,PLT,LSERR,PRTPLT,SCALE2)
- COMMAND LANGUAGE PROCESSOR TREE
- CLP TREE EXECUT-(DMP,SLAM,PLTFIL,EXINTO,PCHCMD,DSKOPN,DSKCLS,RDCCMD,SAVC,MD,TTLCMD,PRGCM,RENAME,DELCMD)
- EXEMPT GLOBAL KWNAM1,KWNAM2,KWNAM3,KWNAM4,KWNAM5,FMTPTR,DEFALT
- EXEMPT GLOBAL DEFLST,ARGTYP,LOCDAT,TMPFIL,ERR,OUTFL,KWTASK
- EXEMPT GLOBAL HOLSTR,ERMODE,ARGTBL,MASK,KEYWRD,MSTRG,DSKNAM,FITS,CHAR
- EXEMPT GLOBAL XQTTBL,SYMDAT,PARTBL,SCNTBL,ANGDAT
- EXEMPT GLOBAL PLTFMT,PLTLAB,PLTLCV,PLTCM1,PLTCM2,PLTCM3,PLTCM4
- GLOBAL SYSTEM COMMON BLOCKS
- EXEMPT GLOBAL Q8.I0.,FCL.C.,PUT.F0,REW.F0,CLSF.F0,OPEN.F0,GET.F0
- EXEMPT GLOBAL CON.RM,A0B.RM
- EXEMPT GLOBAL MEMC.RM,I0.BUF.
- EXEMPT GLOBAL PLTCOM
- END EXEMPT

Table 3
EXEMPT LOGICAL FILES

<u>LFN</u>	<u>DESCRIPTION</u>
TAPE3	Output file from scanning input units TAPE4 and TAPE5 -- this will be the input to the LIP
TAPE4	Alternate user input
TAPE5	User input unless an ALTFIL command has been detected
TAPE6	Output file
TAPE8	Punch file
TAPE67	Input to SFIELD created by the routine WIRANT
TAPE98	METAPLOT output file
TAPE99	Random access data base file

loaded into memory only when needed and, likewise, data are unloaded from memory and placed on the random access system when no longer needed. (For a complete description of the random access system, refer to the EXEMPT Users Manual).

During pass 2, logical operations, SLAM operations, and arithmetic operations are executed. Each user command is executed sequentially by the module designed to interpret the argument list specified. Intramodule communication is done through the symbol table attribute list and other data structures stored in the named common.

When the final task has been completed, control returns to the EXEMPT driver. If a RUN directive had been encountered, the LIP is recalled for the execution of another set of commands. If the LIP is not recalled, the driver completes execution by closing the random access system file.

The use of modular construction allowed EXEMPT to be loaded using the segmentation loader available on the CDC system at the AFWL. Figure 2 illustrates the segments by name. By comparing Figure 2 to Figure 1, one can see how the segments and architecture parallel one another.

Table 1 contains a list of all the subroutines within EXEMPT and Table 2(a) is a subroutine breakdown of EXEMPT by segment. Table 2(b) contains the SEGLOAD directives. The EXEMPT logical file names (LFNs) are presented in Table 3 with a description of their use.

The remainder of this chapter discusses each major section of the architecture, shown in Figure 1, down to level III. Section III gives functional flowcharts down to the subroutine level for each user requested operation.

1. EXEMPT DRIVER DESCRIPTION - LEVEL I

The sequence of operations is shown in the EXEMPT driver functional flowchart which is illustrated in Figure 3.

The first operation of the EXEMPT driver is to call the subroutine PRESCN which reads the user's input up to an end of file (EOF) on LFN TAPES and writes each statement to LFN TAPE3. The function of PRESCN is twofold. First, as each input card image is read, all blank cards and comments are ignored. Also, for those card images that contain more than one statement per image, PRESCN breaks them up so that there is only one statement per card image on unit 3. Secondly, the ALTFIL and SCAN directives are executed in PRESCN.

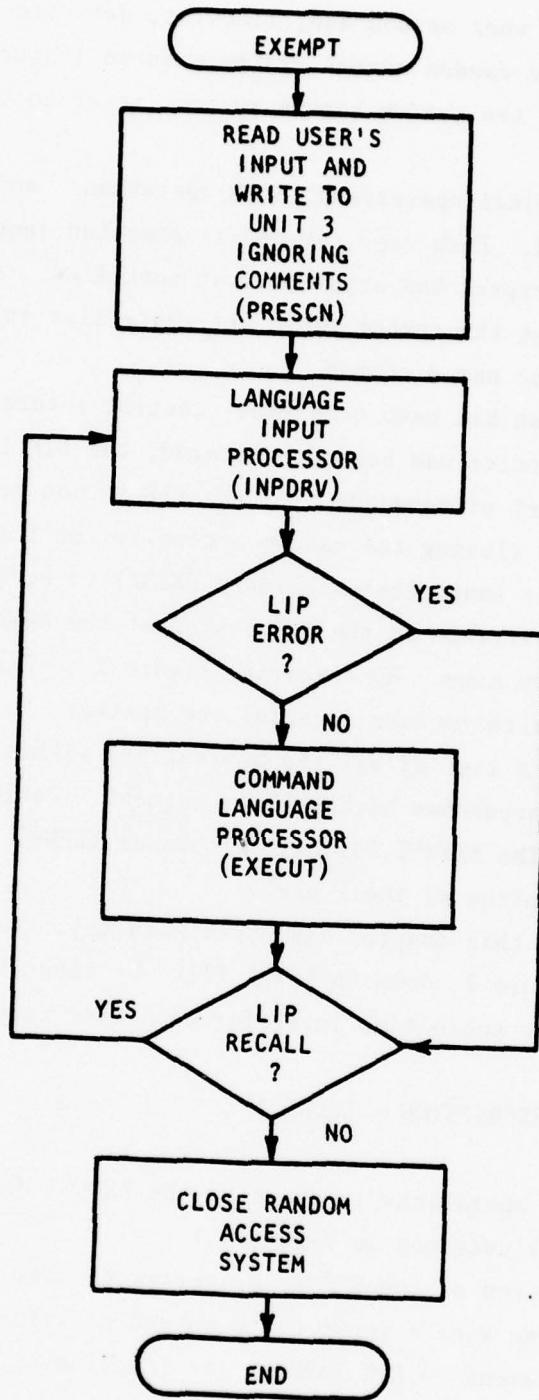


Figure 3. EXEMPT Functional Flowchart

The next operation is to call the subroutine INPDRV which is the main driver for the LIP. If no errors were detected by the LIP, processing continues; otherwise, EXEMPT skips the execution phase and checks to see if the LIP is to be recalled.

The subroutine EXECUT, the main driver for the CLP, is then called for code execution. If the CLP detects an error, control returns to the EXEMPT driver. If the LIP is to be recalled, the same process is repeated; otherwise, the random access file system is closed and EXEMPT is terminated.

When the LIP is to be recalled, all symbol attributes and their associated data remain intact, and all peripheral files that were open remain open. Thus, when the LIP is recalled there is no change to the internal structure or the interface to the host computer.

2. LIP DRIVER - LEVEL II

The LIP is responsible for decoding the user's input into a form that will be more efficient for execution by the CLP. The resultant argument list is a coded list that is explained in more detail in Section II.

The functional flowchart is shown in Figure 4. The first function of the LIP is to initialize particular variables and arrays used by the LIP. These include the argument list, loop counter, and the END and RUN flags. The next operation is the calling of subroutine SCAN whose primary function is that of a lexical analyzer. This is the most basic phase of any translation. The input program is subdivided into its elementary items (i.e., numbers, operator symbols, delimiters, keywords, blanks, etc.). Each item type is identified and each item of a statement has an associative type tag when passed to the next phase of operation, PARSE. This is illustrated in the scanner functional flowchart of Figure 5.

The function of the PARSE subroutine is to identify a string of items and form them into a statement of one of two types. The first type is an arithmetic expression. These are identified when the second field of a statement contains an equal sign. The argument list is built using Reverse Polish Notation (RPN) for arithmetic expressions. Due to the structure of the RPN stack, the equal sign will be the last item placed on the argument list.

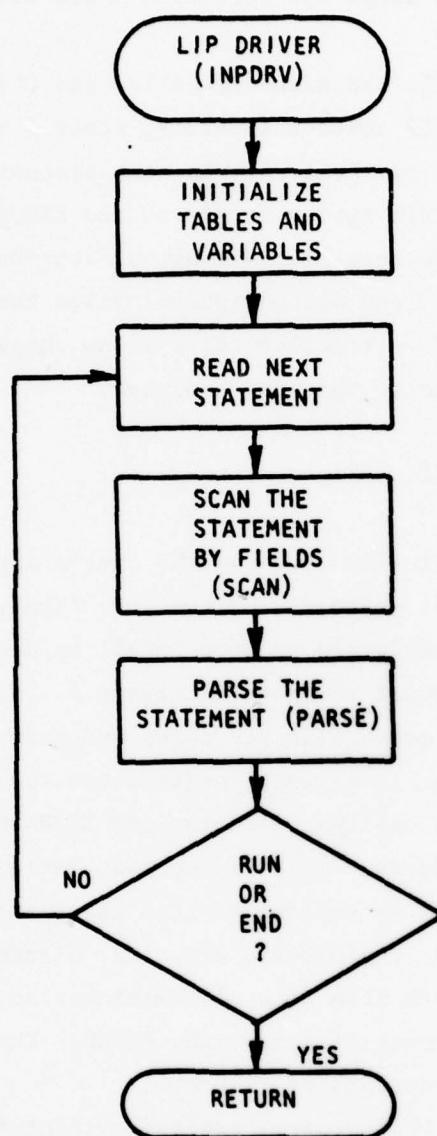


Figure 4. LIP Functional Flowchart

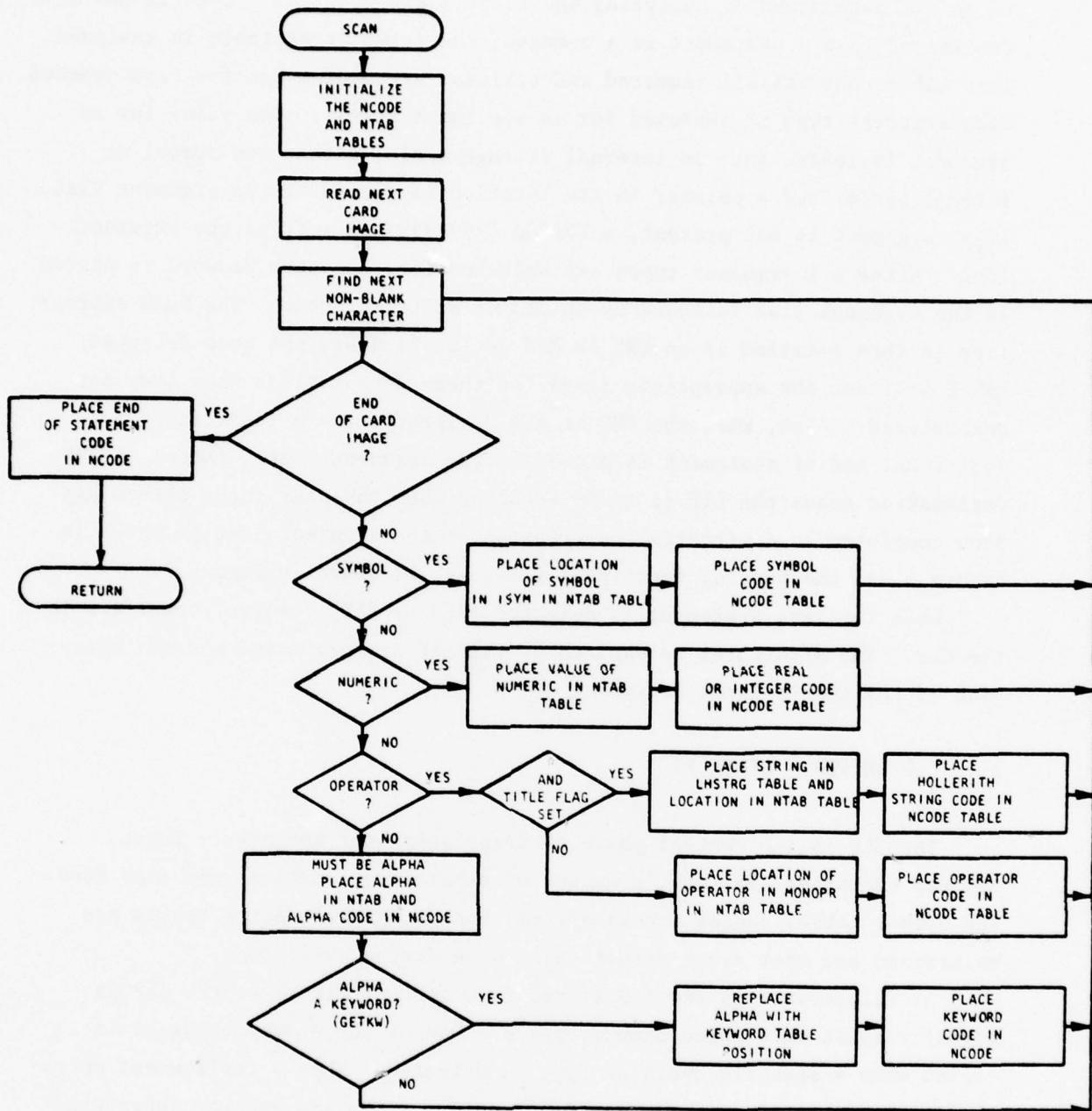


Figure 5. Functional Flowchart of the LIP Scanner

The second type of statement is the command declaration. Command declarations are determined by analyzing the first field position. Once it has been determined that a statement is a command, the format task table is analyzed. This table contains all required and optional argument types for each command. Each argument type is searched for in the input stream. The value for an argument is loaded into an internal storage table, either the symbol or literal table, and a pointer to its location is placed on the argument list. If an argument is not present, a NOPCOD (-999999) is left on the argument list. After all argument types are searched for, the task keyword is placed in the argument list followed by an end of statement code. The SCAN subroutine is then recalled if an END or RUN declaration has not been detected. PARSE will set the appropriate flags for these declarations when they are encountered. Also, when the END or RUN declarations are identified, an additional end of statement is placed on the argument list. (Note: A RUN declaration means the LIP is to be recalled when the user input stream has been completed.) A pictorial description of the argument list is shown in Figure 6 and the parsing functional flowchart is shown in Figure 7.

When the last statement is detected (RUN or END), control transfers to the CLP. The structures recognized by the LIP are processed and the execution of the user's input begins.

3. CLP DRIVER - LEVEL II

The CLP is the central phase of translation for the user's input. Besides executing the code, a number of subsidiary functions are also done. The symbol table, random access system, and peripheral device tables are maintained and most error detection is done during this phase.

As illustrated in the functional flowchart of Figure 8, the CLP is actually split into three subprocessors. One of these subprocessors is called when a specific operator type is detected. When a replacement operator (=) is detected, an arithmetic operation is implied and the subroutine EXPDRV is called to execute the statement. Logical operations and SLAM operations both end with an end of statement code operator. However, the command code for both types of operations is loaded into the argument list just prior to the end of statement so that determining the type of operation is done by polling the argument list entry at this location. The subroutine

EXPRESSION 1	STATEMENT 1
EQUAL SIGN	
ARG1	
ARG2	STATEMENT 2
COMMAND 1	
END OF STATEMENT	
ARG1	STATEMENT 3
COMMAND 2	
END OF STATEMENT	
EXPRESSION 2	STATEMENT 4
EQUAL SIGN	
EXPRESSION 3	
EQUAL SIGN	STATEMENT 5
ARG1	
ARG2	
ARG3	STATEMENT 6
COMMAND 3	
END OF STATEMENT	
END OF STATEMENT	LAST STATEMENT (END OR RUN DECLARATION)

Figure 6. Argument List Table

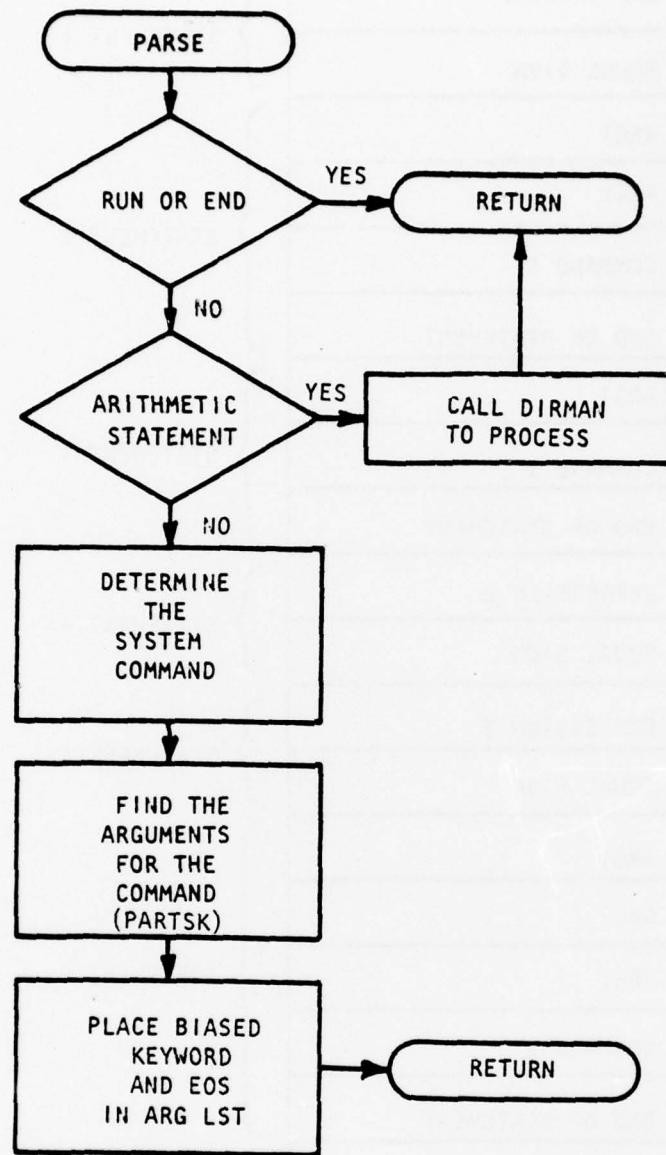


Figure 7. Functional Flowchart of the LIP Parser

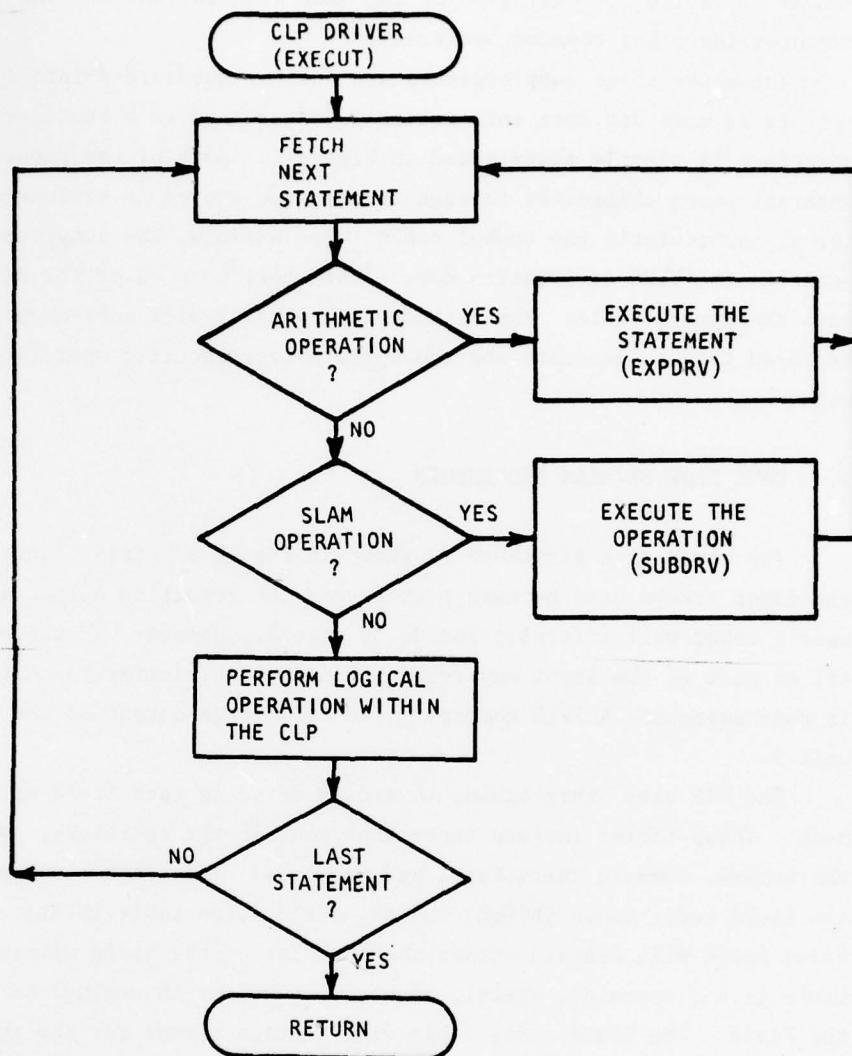


Figure 8. Functional Flowchart of the CLP Driver

SUBDRV is called for execution of the SLAM declarations and the CLP itself executes the other command declarations.

The above three subprocessors are further subdivided into smaller processors as more and more information is determined in a statement. This structure is clearly illustrated in Figure 1. Each of the subprocessors interact among themselves through information stored in various data structures, particularly the symbol table. For example, the subprocessor that performs the TYPE declaration does little more than enter the declared type into the symbol table. The arithmetic processor will subsequently use the declared type to generate the appropriate type-specific operation for the code.

4. DATA FLOW BETWEEN PROCESSORS

The basic data structure is shown in Figure 9. This figure illustrates the input arrays used by each process and the resulting output arrays. The user's input will initially reside on unit 5. However, if the user wishes, all or part of the input may reside on unit 4. Transfer to unit 4 for input is done using the ALTFIL command. The card image output of the prescanner is unit 3.

The LIP uses other tables to aid in decoding each field within a statement. These tables include those that contain the operators, symbols, alpha characters, numeric characters, and keywords. The scanner output arrays are the field codes table (NCODE) and the field value table (NVAL). The field value table will contain either the location of the field within a character table (i.e., operator, symbol, keyword, or Hollerith string) or the value of the field. The field codes table will contain a code for the type of field so that the parser will know whether to look up the value in the character tables or to fetch the value directly from the field table. Table 4 illustrates the above discussion.

The parser uses the NCODE and NVAL tables to decode a statement. The coded result is stored in the argument list table (NARGTB). Also, the parser generates the symbol table (SYMTBL), literal table (LITNUM), and loop table (NLOOPS) as each statement is decoded. Control is then passed to the CLP.

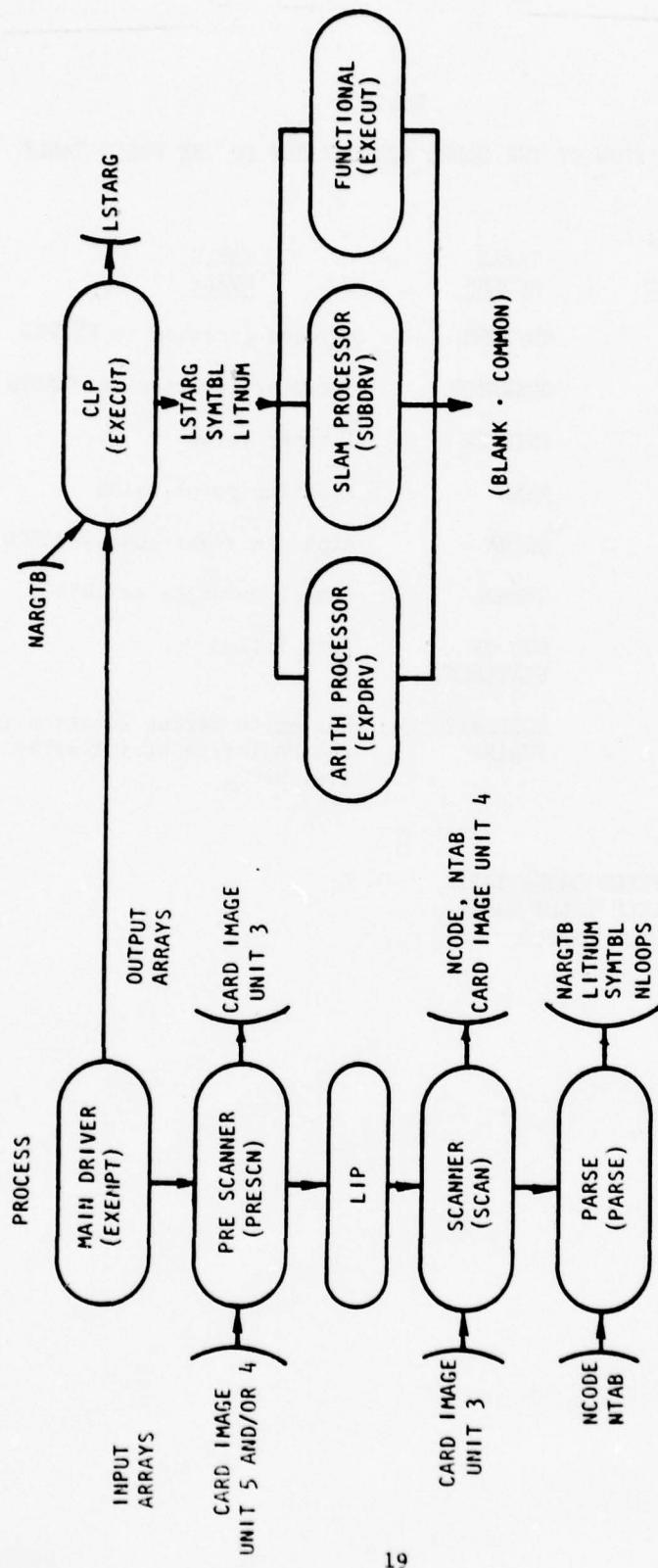


Figure 9. Data Flow Between Processors

Table 4
ASSOCIATION OF THE CODED FIELD TABLE TO THE FIELD TABLE

<u>VALUE</u>	<u>MNEMONIC</u>	<u>TABLE NCODE*</u>	<u>TABLE NVAL+</u>
1	NCODKW	KEYWORD	Keyword location in KEYWRD
2	NCODOP	OPERATOR	Operator location in MONOPR
3	NCODIN	INTEGER	Integer value
4	NCODFP	REAL	Floating point value
5	NCODAL	ALPHA	Alpha in right-justified BCD **
6	NCODSM	SYMBOL	Symbol location in ISYM
7	NCODSE	END OF STATEMENT	Zero filled
8	NCODHS	HOLLERITH STRING	Hollerith string location in the Hollerith string array listing

* NCODE - FIELD CODES TABLE

+ NVAL - FIELD VALUE TABLE

** BINARY CODED DECIMAL

The argument list is extracted from the NARGTB array and transferred to the LSTARG array for subsequent use by the subprocessors. This is done to maintain the integrity of the NARGTB array from pass 1 to pass 2. The argument list table (LSTARG) passed to each major driver subroutine from the CLP is actually only one statement pulled from the master argument list table (NARGTB). The shortened argument list, symbol table, and literal table all interact to generate data in the array BLNKCM before the data are stored in the random access system.

Section II gives in-depth discussion of the common blocks containing the data structures. Also included are functional flowcharts of each command available within the present language and the functional flowcharts of the arithmetic operations.

SECTION II

DATA STRUCTURES

Of importance to every language is the linkage of each of its processor modules through data structures. The data structures are a collection of items contained in several tables which are related to each other in some fashion. These tables and the common blocks containing them are discussed in the following sections. Each table briefly discusses its architecture and each common block is fully described by including a definition of each variable contained.

1. COMMON BLOCK GLOSSARY (Variable Name and Common Location)

AXISX	- *	PLTFMT	IDECSR	- *	ERMODE
AXISY	- *	PLTFMT	IDIG	- *	CHARTR
CALCP	- *	PLTCM1	IDPCOM	- *	PLTCM4
CHAR	- *	CHARTR	IDPLT	- *	EHR
COSA	- *	PLTCM3	IDUMP	- *	EHR
CW	- *	PLTCM2	IEOS	- *	XQTTBL
DBGPR	- *	SCNTBL	IEQUAL	- *	SCNTBL
DFET	- *	MSTRG	IERCOD	- *	ERMODE
ENORM	- *	ANGDAT	IERROR	- *	ERMODE
ERRSCN	- *	SCNTBL	IFILID	- *	DSKNAM
FACTE	- *	ANGDAT	IFITW	- *	FITS
FACTH	- *	ANGDAT	IFITWO	- *	FITS
FIRST	- *	SCNTBL	IFINOW	- *	DSKNAM
FLTLIT	- *	PARTBL	IGNORE	- *	SCNTBL
FMAX	- *	ANGDAT	IL	- *	PLTCM3
FMIN	- *	ANGDAT	ILLCHR	- *	ERMODE
FRQFIL	- *	DEFLST	IMESG	- *	PLTCM2
FRQMAX	- *	DEFLST	IPAREN	- *	SCNTBL
FRQMIN	- *	DEFLST	IPASS	- *	XQTTBL
FULFRM	- *	PLTLGV	IPEROD	- *	SCNTBL
GREAL	- *	DEFLST	IPGNUM	- *	PLTCM2
HL	- *	PLTCM3	IPPLUS	- *	SCNTBL
HLB	- *	PLTCM1	ISEGMENT	- *	ANGDAT
HML	- *	PLTCM1	ISOFF	- *	SCNTBL
HNO	- *	PLTCM1	ISON	- *	SCNTBL
HSURF	- *	ANGDAT	ISTMT	- *	SCNTBL
IALPH	- *	CHARTR	ISTOR	- *	SYMTBL
IALPHA	- *	SCNTBL	ISYM	- *	CHARTR
IBLANK	- *	SCNTBL	ISYMS	- *	SYMTBL
IBLKCD	- *	SCNTBL	ITYPED	- *	LOCDAT
IBLKCM	- *	BLKCM	IUN	- *	MSTRG
IBLKNM	- *	DSKNAM	IUNDX	- *	MSTRG
IBLNK	- *	PLTCM2	IWARN	- *	ERMODE
IBYTSZ	- *	SCNTBL	IZERO	- *	SCNTBL
ICHAR	- *	PLTCM3	JFET	- *	TMPFIL
ICOND	- *	ERMODE	JFILE	- *	TMPFIL
IDANGR	- *	ERMODE	JNUM	- *	TMPFIL
IDD	- *	DSKNAM	JUNIT	- *	TMPFIL

KOLATT	- • SYMTBL	KWDROP	- • KWNAM1
KOLCNT	- • PARTBL	KWD1	- • KWNAM5
KOLCOD	- • PARTBL	KWD2	- • KWVAM5
KOLFAM	- • SYMTBL	KWD3	- • KWNAMS
KOLLBL	- • PARTBL	KWEDIT	- • KWNAM2
KOLLOC	- • SYMTBL	KWEND	- • KWNAM1
KOLNAM	- • SYMTBL	KWENGI	- • KWNAM3
KOLNPT	- • SYMTBL	KWEOF	- • KWNAM1
KOLTIM	- • PARTBL	KWEPS	- • KWNAMS
KOLTSK	- • PARTBL	KWERR	- • ERMODE
KOLTYP	- • SYMTBL	KWEXP	- • KWNAM2
KOLVAL	- • PARTBL	KWFILE	- • KWNAM1
KPRNCT	- • PARTBL	KWFILF	- • KWNAMS
KTEMOT	- • PARTBL	KWFMAX	- • KWNAMS
KURSQF	- • TMPFIL	KWFMIN	- • KWNAMS
KWA	- • KWNAM5	KWFMTP	- • FMTPTR
KWA8S	- • KWNAM2	KWFNND	- • XQTTBL
KWAEFF	- • KWNAMS	KWFNST	- • XQTTBL
KWALFL	- • KWNAM1	KWFORM	- • KWNAM1
KWALOG	- • KWNAM2	KWFQFL	- • KWNAM4
KWALPH	- • KWNAMS	KWFQMN	- • KWNAM4
KWANGL	- • KWNAMS	KWFQMX	- • KWNAM4
KWARGS	- • ARGTBL	KWFT	- • KWNAM2
KWAS19	- • KWNAM3	KWFUN	- • KWNAM1
KWATAN	- • KWNAM2	KWF1	- • KWNAMS
KWATN2	- • KWNAM2	KWGETR	- • KWNAM1
KWAVBA	- • KWNAM3	KWHEIG	- • KWNAMS
KWAVGS	- • KWNAM3	KWHFAN	- • KWNAM3
KWA1	- • KWNAMS	KWINDC	- • KWNAM3
KWA2	- • KWNAM5	KWINDP	- • KWNAM4
KWB	- • KWNAM5	KWINTP	- • KWNAM2
KWC	- • KWNAM5	KWINVT	- • KWNAM2
KWCALL	- • KWNAM1	KWLBEL	- • KWNAM1
KWCMPX	- • KWNAM2	KWLENG	- • KWNAMS
KWCNVT	- • KWNAM2	KWLGLG	- • KWNAM4
KWCOAX	- • KWNAM3	KWLGLN	- • KWNAM4
KWCOCK	- • KWNAM3	KWLIN	- • KWNAM4
KWCONO	- • KWNAM3	KWLST	- • KWNAM1
KWCONJ	- • KWNAM2	KWLMT	- • PARTBL
KWCOS	- • KWVAM2	KWLNLG	- • KWNAM4
KWCOUF	- • KWNAM3	KWLNLN	- • KWNAM4
KWCURF	- • KWNAM4	KWLOG	- • KWNAM4
KWC0	- • KWNAMS	KWLOOP	- • KWNAM1
KWC1	- • KWNAM5	KWL1	- • KWNAMS
KWC2	- • KWNAM5	KWL2	- • KWNAMS
KWC3	- • KWNAM5	KWL3	- • KWNAMS
KWDB	- • KWNAM4	KWL4	- • KWNAMS
KWDBCB	- • KWNAM3	KWMARB	- • KWNAM3
KWDBC1	- • KWNAM3	KWMAX	- • KWVAM2
KWDBLG	- • KWNAM4	KWMEGE	- • KWNAM2
KWDBLN	- • KWNAM4	KWMIN	- • KWNAM2
KWDBUG	- • KWNAM1	KWMNVL	- • KWNAM4
KWDCFL	- • KWNAM4	KWMOD	- • KWNAM2
KWDELE	- • KWNAM1	KWMODE	- • KWNAMS
KWDELT	- • KWNAMS	KWMP	- • KWNAM4
KWDEPT	- • KWNAM5	KWMXVL	- • KWNAM4
KWDGRS	- • KWNAM4	KWM1	- • KWNAMS
KWDIV	- • KWNAM5	KWNAME	- • KEYWRD
KWDLFQ	- • KWNAM4	KWNDEX	- • KWNAM4
KWDLTM	- • KWNAM4	KWNPTS	- • KWNAM4
KWDM	- • PARTBL	KWNXT	- • KWNAM4

KWN1	- • KWNAM5	KWSKPB	- • KWNAM1
KWN2	- • KWNAM5	KWSKPF	- • KWNAM1
KWOFF	- • KWNAM4	KWSNGL	- • KWNAM3
KWON	- • KWNAM4	KWSPCM	- • KW4AM4
KWPAGL	- • KWNAM4	KWSQRT	- • KWNAM2
KWPAGW	- • KWNAM4	KWSVFN	- • KWNAM4
KWPART	- • KWNAM4	KWTAN	- • KWNAM2
KWPFI	- • KWNAM5	KWTAU	- • KWNAM5
KWPITO	- • KWNAM3	KWTDEP	- • KWNAM2
KWPLOT	- • KWNAM1	KWTDSN	- • KWNAM2
KWPNLJ	- • KWNAM3	KWTHT	- • KWNAM5
KWPOLY	- • KWNAM2	KWTITE	- • KWNAM1
KWPRPT	- • KWNAM1	KWTLEX	- • KWNAM4
KWPT1	- • KWNAM4	KWTLEY	- • KWNAM4
KWPT2	- • KWNAM4	KWTMFL	- • KWNAM4
KWPUCH	- • KWNAM1	KWTMMN	- • KWNAM4
KWPURG	- • KWNAM1	KWTMMX	- • KWNAM4
KWRADS	- • KW4AM4	KWTST8	- • KWTASK
KWRDAL	- • KWNAM3	KWTYPE	- • KWNAM1
KWRDAR	- • KWNAM3	KWVPCM	- • KWNAM4
KWREAC	- • KWNAM1	KWWEAB	- • KWNAM3
KWREAD	- • KWNAM1	KWWEBA	- • KWNAM3
KWREDF	- • KWNAM1	KWWEIG	- • KWNAMS
KWREP	- • KWNAM4	KWWIDT	- • KWNAM5
KWREWD	- • KW4AM1	KWWIRE	- • KWNAM5
KWRI	- • KWNAM4	KWWROO	- • KWNAM3
KWRITE	- • KWNAM1	KWWWEL	- • KWNAM3
KWRNAM	- • KWNAM1	KWWWGN	- • KW4AM3
KWRUN	- • KWNAM1	KWZERO	- • KWNAM4
KWRVBE	- • KW4AM3	KWZO	- • KWNAM5
KWR1	- • KW4AM4	KWZS	- • KWNAM5
KWR10	- • KWNAM4	KWZT	- • KWNAM5
KWR11	- • KW4AM4	KWZ0	- • KWNAM5
KWR12	- • KWNAM4	LAB	- • PLTLAB
KWR13	- • KWNAM4	LABPAG	- • PLTLGV
KWR14	- • KWNAM4	LABVRT	- • PLTLGV
KWR15	- • KWNAM4	LABX	- • PLTLAB
KWR16	- • KW4AM4	LABY	- • PLTLAB
KWR17	- • KW4AM4	LCUTMK	- • PLTLGV
KWR18	- • KWNAM4	LHSTRG	- • HOLSTR
KWR19	- • KWNAM4	LIBEXM	- • FMTPTR
KWR2	- • KWNAM4	LINLOG	- • SCNTBL
KWR20	- • KWNAM4	LIST	- • ANGDAT
KWR3	- • KWNAM4	LISTDM	- • DSKVAM
KWR4	- • KWNAM4	LISTRE	- • DSKVAM
KWRS	- • KWNAM4	LITNUM	- • PARTBL
KWR6	- • KWNAM4	LKWFUC	- • PARTBL
KWR7	- • KW4AM4	LLAST	- • SCNTBL
KWR8	- • KWNAM4	LLIMKW	- • PARTBL
KWR9	- • KWNAM4	LLIN	- • SCNTBL
KWSAFE	- • KWNAM4	LLOG	- • SCNTBL
KWSAVE	- • KW4AM1	LINDEX	- • PARTBL
KWSAVR	- • KWNAM1	LNK	- • LOCDAT
KWSCAN	- • KWNAM1	LOCBLK	- • CHARTR
KWSFLD	- • KWNAM1	LOCCom	- • CHARTR
KWSGLE	- • KWNAM3	LOCOD	- • LOCDAT
KWSHT	- • KW4AM2	LOCDIV	- • CHARTR
KWSHIE	- • KWNAM3	LOCDOl	- • CHARTR
KWSIGM	- • KW4AM5	LOCEQU	- • CHARTR
KWSIGN	- • KWNAM2	LOCEXP	- • CHARTR
KWSIN	- • KWNAM2	LOCHS	- • HOLSTR

LOCITP	- •	CHARTR	NCODFP	- •	SCNTBL
LOCMIN	- •	CHARTR	NCODHS	- •	SCNTBL
LOCMUL	- •	CHARTR	NCODIN	- •	SCNTBL
LOCPER	- •	CHARTR	NCODKW	- •	SCNTBL
LOCPLS	- •	CHARTR	NCODOP	- •	SCNTBL
LOCRTP	- •	CHARTR	NCODPE	- •	SCNTBL
LOGERR	- •	PLTCM2	NCODSE	- •	SCNTBL
LSQNUM	- •	PLTLGV	NCODSM	- •	SCNTBL
LSTACK	- •	PARTBL	NE	- •	ERR
LSTARG	- •	XQTTBL	NENDFG	- •	PARTBL
LSTART	- •	SCNTBL	NEQUAL	- •	XQTTBL
LTITLE	- •	HOLSTR	NFIELD	- •	ERMODE
LUNMPF	- •	PLTCM2	NFTNLB	- •	FMTPTR
LUNPIF	- •	PLTCM2	NFUCHD	- •	PARTBL
LUPRNT	- •	SCNTBL	NFUNFG	- •	PARTBL
LUREAD	- •	SCNTBL	NFWBUF	- •	FITS
MAGRI	- •	LOCDAT	NKEYWD	- •	PARTBL
MATCH	- •	PARTBL	NLOOPS	- •	PARTBL
MAXALP	- •	CHARTR	NOGOFG	- •	XQTTBL
MAXCML	- •	SCNTBL	NOMTCM	- •	PARTBL
MAXCHR	- •	ERMODE	NOPCOD	- •	PARTBL
MAXCOL	- •	SYMTBL	NOZL	- •	PLTLGV
MAXFPT	- •	CHARTR	NPARGL	- •	PARTBL
MAXHSL	- •	HOLSTR	NPDATA	- •	PARTBL
MAXINT	- •	CHARTR	NPFLIB	- •	FMTPTR
MAXKW	- •	KEYWRO	NPLITN	- •	PARTBL
MAXLIB	- •	FMTPTR	NPLOOP	- •	PARTBL
MAXNUM	- •	CHARTR	NPRCNT	- •	PARTBL
MAXOPR	- •	CHARTR	NPRSTK	- •	PARTBL
MAXPWR	- •	CHARTR	NRUNFG	- •	PARTBL
MAXSCN	- •	SCNTBL	NSCOL	- •	SCNTBL
MAXSTK	- •	ERMODE	NTAB	- •	SCNTBL
MAXSTR	- •	ERMODE	NTABSV	- •	SCNTBL
MAXSYM	- •	CHARTR	NUMARG	- •	XQTTBL
MAXVAL	- •	XQTTBL	NUMBCM	- •	DEFLST
MDECX	- •	PLTCM1	NUMCHR	- •	HOLSTR
MDECY	- •	PLTCM1	NUMD	- •	LOCDAT
MKMX	- •	PARTBL	NUMDEC	- •	CHARTR
MNSUBC	- •	PARTBL	NUMOPS	- •	LOCDAT
MODE	- •	ANGDAT	NUND	- •	MSTRG
MONOPR	- •	CHARTR	NUPBKW	- •	PARTBL
MSTNDX	- •	MSTRG	NUPKWF	- •	PARTBL
MXARGT	- •	PARTBL	NVAL	- •	SCNTBL
MXFPCT	- •	ERMODE	NVALMX	- •	PARTBL
MXHSTR	- •	HOLSTR	NXFMT	- •	PLTLAB
MXI	- •	PLTCM1	NX1OPW	- •	PLTLGV
MXINCT	- •	ERMODE	NYFMT	- •	PLTLAB
MXMAT	- •	PARTBL	PAGEX	- •	PLTFMT
MXPRCT	- •	ERMODE	PAGEY	- •	PLTCM1
MXSUBC	- •	PARTBL	PAGLEN	- •	DEFLST
MXSYM	- •	SYMTBL	PAGWID	- •	DEFLST
MXSYSC	- •	PARTBL	PDAT	- •	PLTCM1
MYI	- •	PLTCM1	PHI	- •	ANGDAT
NARGTB	- •	PARTBL	PID	- •	PLTCM1
NARGTP	- •	PARTBL	PLTCLO	- •	PLTLGV
NBIAS	- •	PARTBL	PLTERR	- •	PLTLAB
NCARD	- •	SCNTBL	PLTING	- •	PLTLGV
NCODAL	- •	SCNTBL	PLTLBL	- •	PLTLGV
NCODE	- •	SCNTBL	PLTOPN	- •	PLTLGV
NCODFN	- •	SCNTBL	PNV	- •	ANGDAT

PRINT	- •	DSKNAM
PTIM	- •	PLTCM1
RDERR	- •	ERR
SFNX	- •	PLTCM4
SHAD	- •	ANGDAT
SINA	- •	PLTCM3
SPCM	- •	DEFLST
SPX	- •	PLTFMT
SPY	- •	PLTFMT
STPFLG	- •	PLTLGV
SV	- •	ANGDAT
SYMDAT	- •	SYMTBL
SYMSTR	- •	SYMTBL
TAU	- •	ANGDAT
THETA	- •	ANGDAT
TIMFIL	- •	DEFLST
TIMMAX	- •	DEFLST
TIMMIN	- •	DEFLST
TML	- •	PLTCM1
USRCOM	- •	SCNTBL
VAL	- •	SCNTBL
VPCM	- •	DEFLST
WIRE	- •	ANGDAT
XMAX	- •	PLTCM4
XMIN	- •	PLTCM4
XP	- •	PLTCM3
XPSF	- •	PLTCM2
XS	- •	PLTCM4
YMAX	- •	PLTCM4
YMIN	- •	PLTCM4
YP	- •	PLTCM3
YPSF	- •	PLTCM2
YS	- •	PLTCM4
ZDEBUG	- •	XQTTBL
ZERO	- •	DEFLST
ZRAND	- •	XQTTBL

2. MAJOR DATA STRUCTURES

The LIP requires the use of three tables for generating the object code for the CLP. They are the symbol table, the literal table, and the argument list table. The LIP loads the argument list table with the following information: codes for operators or keywords, pointers to the symbol table, and pointers to the literal table. The symbol and literal tables contain pointers to other tables as well as information pertinent to the execution of the object code. Figure 10 illustrates the table linkage.

Different ranges of numeric values have special meaning when in the argument list. They are:

<u>Value</u>	<u>Meaning</u>
0	None
1-1000	Pointer to the symbol table
1001	Code for the operator =
1002	Code for the operator +
1003	Code for the operator -
1004	Code for the operator *
1005	Code for the operator /
1006	Code for the operator **
1007	Code meaning an end of statement
>1007	Code for a command declaration
-999999	Code meaning a null operation
-1 to -100	Pointer to the literal table

The symbol table is one of the central data structures. During the LIP, all symbols are placed in the symbol table in right-justified, zero-filled BCD. No other entries are made to the symbol table in the LIP. The CLP enters all other information into the symbol table as it processes the logical, SLAM, and arithmetic operations.

The literal table's primary purpose is to contain the constants (real, integer, or alphanumeric) referenced throughout the user's input. Other secondary information is also contained within the literal table. These are the pointers to the Hollerith string table, the keyword table, and the symbol table. The information contained in the literal table completes the linkage of the required data structures.

The three tables illustrated in Figure 10 are contained in the named common blocks /PARTBL/, /SCNTBL/, and /SYMDAT/. These common blocks are discussed in the following subsections and include a functional definition for each variable contained in the common blocks.

a. Common /SYMDAT/

```

LEVEL 2,SYMTBL,MXSYM,MAXCOL,KOLNAM,KOLLOC,KOLFAM,KOLTYP,KOLNPT,
      KOLATT,SYMSTR,ISTOR
COMMON/SYMDAT/ SYMTBL(100,6), MXSYM, MAXCOL, KOLNAM, KOLLOC,
      KOLFAM, KOLTYP, KOLNPT, KOLATT, SYMSTR(100),
      ISTOR
DIMENSION ISY4ST(100)
EQUIVALENCE (SYMSTR(1),ISY4ST(1))
INTEGER SYMTBL

```

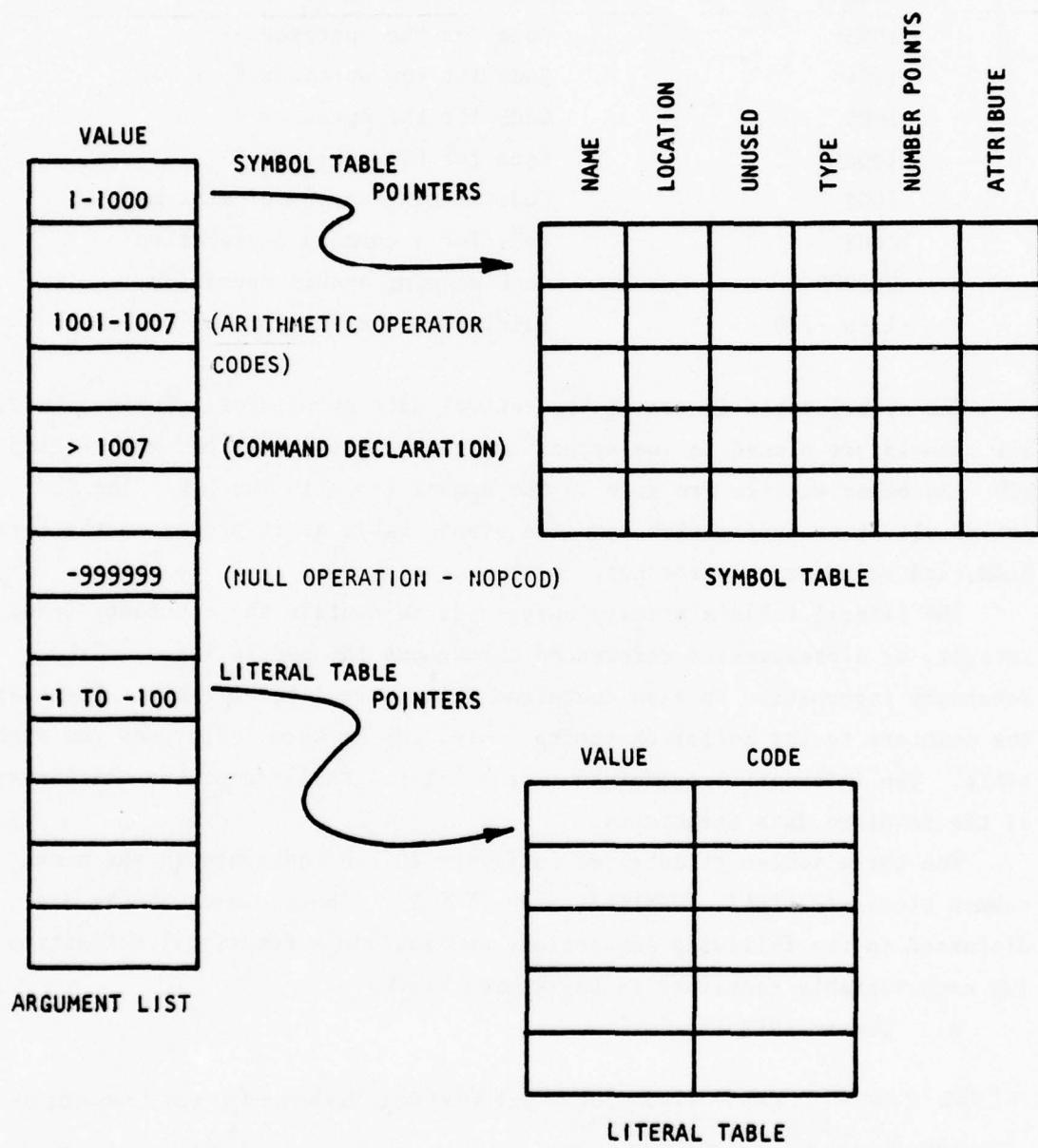


Figure 10. Major Data Structure Linkage

EXEMPT maintains a symbol table which contains the name and attributes of all symbols referenced in the user's ECL. The symbol table is the array SYMBTL which is stored in the named common /SYMDAT/. The current attributes, their columns, and the attribute meanings for data sets containing more than one element are presented in Table 5.

For those symbols which are single element data sets, the named common /SYMDAT/ has the following meaning:

KOLLOC - The storage indicator will be negative when KOLNPT is set to 1. The absolute value of KOLLOC is the index of the symbol's value in SYMSTR.

KOLTYP - Will be set to zero (0).

KOLNPT - Will be set to one (1).

The attribute word KOLATT has the following meaning for all data sets:

Bits 0-1 - Format type
00 - not set
01 - real data set
10 - integer data set
11 - complex data set

Bits 2-3 - Complex type
00 - should be real data set
01 - real-imaginary
10 - magnitude-phase
11 - decibel-phase

Bits 4 - Phase type
0 - radians
1 - degrees

Bits 5-59 - Unused

Table 5
SYMDAT ENTRIES

<u>VALUE</u>	<u>SYMBOL</u>	<u>FUNCTION</u>
1	KOLNAM	Data set name, right-justified BCD
2	KOLLOC	Storage indicator, file, and record (Bits 0-29 and 30-59, respectively)
3	KOLFAM	Family attribute (unused)
4	KOLTYP	Type of data, positive for frequency data, negative for time data. Absolute value of KOLTYP points to associated independent data set entry in SYMTBL.
5	KOLNPT	Number of points in the data set
6	KOLATT	Attribute word, described below
100	MXSYM	Row dimension of SYMTBL. Maximum number of SYMTBL entries is expandable to 1000
6	MAXCOL	Column dimension of SYMTBL
N/A	SYMTBL	Two-dimensional array containing symbol names and attributes
N/A	SYMSTR	One-dimensional array containing a value for a one-dimensional data set
N/A	ISTOR	Current pointer-index to SYMSTR

b. Common /PARTBL/

```
COMMON/PARTBL/      LITNUM(100,2), NLOOP(100,4), KOLCOD,
,                  NARGTB(1000), NPRSTK(100), KWDM(7), NARGTP(4),
,                  KOLVAL, KOLLBL, KOLTIM, KOLTSK, NPRCNT, MXSYSC,
,                  MKMX, KWLMT, NVALMX, NUPKWF, LKWFUC, NUPBKW,
,                  LLIMKW, NPARGL, NPLITN, NPDATA, NPLOOP, NBIAS,
,                  KOLCNT, MXARGT, MXMAT, KTEMDT, MATCH, NOMTCH,
,                  NKEYWD, NOPCOD, NFUNFG, NRUNFG, NENDFG, NFUCHD,
,                  KPRNCT, LSTACK(10), LINDEX, MNSUBC, MXSUSC
DIMENSION FLTLIT(100,2)
EQUIVALENCE (LITNUM(1,1),FLTLIT(1,1))
```

Common block PARTBL contains the literal table, argument list table, and loop table. Table 6 contains the current attributes and their columns as it pertains to the literal table. Likewise, Table 7 pertains to the argument list table and Table 8 pertains to the loop table. Table 9 contains those entries used when parsing a statement.

Table 6
PARTBL ENTRIES PERTAINING TO THE LITERAL TABLE

<u>VALUE</u>	<u>SYMBOL</u>	<u>FUNCTION</u>
1	KOLCOD	Column in literal table for the literal type
2	KOLVAL	Column in literal table for the literal value
N/A	NPLITN	Current index of the literal table
N/A	FLTLIT	Literal table of floating point values
N/A	LITNUM	Literal table

Table 7
PARTBL ENTRIES PERTAINING TO THE ARGUMENT LIST TABLE

<u>VALUE</u>	<u>SYMBOL</u>	<u>FUNCTION</u>
7	MKMX	Maximum number of operators including an end of statement
N/A	NARGTB	Argument list table
1000	NBIAS	Biased value that is added to the value of an operator
N/A	NFUCHD	Contains the current biased value of an EXEMPT library function
-999999	NOPCOD	Code for a null or empty field
1000	NPARGL	Dimension of NARGTB

Table 8
PARTBL ENTRIES PERTAINING TO THE LOOP TABLE

<u>VALUE</u>	<u>SYMBOL</u>	<u>FUNCTION</u>
1	KOLLBL	Loop label in right-justified BCD
2	KOLTIM	Number of times a loop is to be executed
3	KOLTSK	The index in the argument list table where the loop begins
4	KOLCNT	Number of times loop has been executed
N/A	NLOOPS	Loop table
N/A	NPLOOP	Current index of the loop table

Table 9
PARTBL ENTRIES

<u>SYMBOL</u>	<u>FUNCTION</u>
KRPRNCT	Counter for the number of right parentheses in a statement
KTEMDT	Contains the name of a temporary data set
KWDM	Not used
KWLMT	Maximum number of keywords
LKWFUC	Maximum number of arithmetic function keywords
LLIMKW	Number of keywords with a task in the task table
LNDEX	Current index of the task table
LSTACK	Not used
MATCH	Flag that when set indicates a symbol has been matched on the symbol table
MNSUBC	Index to first SLAM function in the keyword table
MXARGT	Maximum number of logical command functions
MXMAT	Number of entries in NARGTP
MXSUBC	Index to last SLAM function in the keyword table
MXSYSC	Index to last logical command function in keyword table
NARGTP	Argument table pointer, points to symbol, literal, keyword, and operator tables
NENDFG	Flag that when set means an END declaration
NFUNFG	Flag that when set means an arithmetic function
NKEYWD	Contains current index to keyword table
NOMTCH	Flag that when set means a symbol match was not made in the symbol table
NPDATA	Current index to the symbol table

Table 9
PARTBL ENTRIES (Concluded)

<u>SYMBOL</u>	<u>FUNCTION</u>
NPRCNT	Current index to the operator stack
NPRSTK	Operator stack
NRUNFG	Flag that when set means a RUN declaration found
NUPBKW	Upper bound of keywords in the keyword table
NUPKWF	Upper bound of keywords with an associated task
NVALMX	Maximum number of fields that can be parsed

c. Common /SCNTBL/

```
COMMON/SCNTBL/NCODE(256),NVAL(256),NCOD<W,NCODOP,NCODIN,NCODFN,  
NCODFP,NCODAL,NCODSM,NCODSE,NCODHS,NCODPE,NCARD(81),  
IBYTSZ,LSTART,LLAST,MAXCHL,IALPHA,IZERO,IPLUS,IPAREN,  
IEQUAL,IBLANK,IPEROD,ISON,ISOFF,FIRST,NTABSV,NSCOL,  
NTAB,MAXSCN,LUPRNT,LUREAD,LINLOG,LLIN,LLOG,IBLKCD,  
IGNORE,ERRSCN,USRCOM,DBGPRT,SCNDBG,SCNLIP,ISTMT  
LOGICAL DBGPRT,SCNDBG,SCNLIP  
LOGICAL ERRSCN,USRCOM  
LOGICAL IGNORE  
LOGICAL FIRST  
DIMENSION VAL(256)  
EQUIVALENCE (NVAL(1),VAL(1))
```

Although common block SCNTBL does not contain any of the tables mentioned in the previous section, it does contain the intermediate tables, NCODE and NVAL, and the codes that have parsing and executing mnemonics.

Table 10 is the table of mnemonics and their values for this common block.

When scanning input data, two tables (NCODE and NVAL) are built. The NCODE table contains the code for the type of field (i.e., operator, numeric, etc.) and the NVAL table contains the field (reference table 4) associated with the code. After an entire statement has been placed in the above tables by fields, the statement is ready to be parsed.

3. SUPPORT DATA STRUCTURES

Support data structures include those that maintain the tables for the random and sequential files, Hollerith strings, default values, and error diagnostics to name a few. These structures and tables keep track of the data flow between modules and the correctness of the user's code. The following subsections contain a brief description of each common block containing the data structures.

Table 10
SCNTBL ENTRIES

<u>VALUE</u>	<u>SYMBOL</u>	<u>FUNCTION</u>
.FALSE.	DBGPRT	Printer debugging test from the LIP and CLP when set .TRUE.
.FALSE.	ERRSCN	When set .TRUE. then an LIP error has occurred while scanning
.TRUE.	FIRST	First card of a run stream
IRA	IALPHA	First legal character in display code
IRB	IBLANK	A blank in display code
N/A	IBLAKCD	Not used
6	IBYTSZ	Number of bits a character uses in one word
IR=	IEQUAL	An equal sign in display code
.FALSE.	IGNORE	When set .TRUE. then READC has called SCAN for coded read
IR(IPAREN	Left parenthesis in display code
IR.	IPEROD	Period in display code. Also, last allowed display code character
IR+	IPLUS	Plus sign in display code
0	ISOFF	Used as a test for zero
1	ISON	Used as a test for one
N/A	ISTMT	Number of statements that have gone through the LIP in a run stream
1R0	IZERO	A zero in display code
0	LINLOG	When set to 1 indicates a LIN, LOG, NXT, or REP keyword in point field
3RLIN	LLIN	Keyword LIN in display code
3RLOG	LLOG	Keyword LOG in display code

Table 10
SCNTBL ENTRIES (Continued)

<u>VALUE</u>	<u>SYMBOL</u>	<u>FUNCTION</u>
1	LSTART	First column of card image to be scanned
6	LUPRNT	Output buffer
3	LUREAD	Unit that contains card image
10	MAXCHL	Maximum number of characters in a user defined name
256	MAXSCN	Maximum number of fields to be scanned
N/A	NCARD	Array containing the card image
5	NCODAL	Indicates field is an alpha
N/A	NCODE	Coded field table
N/A	NCODFN	Not used
4	NCODFP	Indicates field is a real number in the NVAL table
8	NCODHS	Indicates field is a Hollerith string in the NVAL table
3	NCODIN	Indicates field is an integer number in the NVAL table
1	NCODKW	Indicates field is a keyword in the NVAL table
2	NCODOP	Indicates field is an operator in the NVAL table
N/A	NCODPE	Not used
7	NCODSE	Indicates end of statement in the NVAL table
6	NCODSM	Indicates field is a symbol in the NVAL table
N/A	NSCOL	Current card image column being scanned
N/A	NTAB	Current field in field tables

Table 10
SCNTBL ENTRIES (Concluded)

<u>VALUE</u>	<u>SYMBOL</u>	<u>FUNCTION</u>
N/A	NTABSV	Numbers of fields scanned in last statement
N/A	NVAL	Field table
.FALSE.	SCNDBG	When set .TRUE. causes debugging text to be batched to output
.FALSE.	SCNLIP	When set .TRUE. causes debugging text to be batched to output
.FALSE.	USRCOM	When .TRUE. a MYCOM declaration is being used
N/A	VAL	Contains floating point fields of the field table

a. Common /ANGDAT/

```
COMMON/ANGDAT/THETA,PHI,TAU,HSURF(3),ENORM(3),MODE,WIRE(5),FACTH,
           FACTE,SHAD,FMAX,FMIN,LIST
DIMENSION ISEGMT(5),SV(3),PNV(3)
INTEGER WIRE
EQUIVALENCE(WIRE(1),ISEGMT(1)),(HSURF(1),SV(1)),(ENORM(1),PNV(1))
LOGICAL LIST
```

Common block ANGDAT contains information necessary for the execution of the routine SFIELD. For a complete description of each variable, reference the SLAM User's Manual.

b. Common /ARGTBL/

COMMON/ARGTBL/KWARGS(256)

The main function of common block ARGTBL is to provide the information needed for determining the type of argument(s) that are associated with each command. The types of arguments may be symbols, defined symbols, numbers, etc., as defined in the keyword task table (KWTSTB). Table 11 is a list of the types of arguments defined.

Table 11
TYPES OF ARGUMENTS

<u>Value</u>	<u>Argument Type</u>
-1	Symbol
-2	Defined Symbol
-3	Symbol or Defined Symbol
-4	Literal
-5	Symbol or Literal
-6	Defined Symbol or Literal
-7	Integer Number
-8	Symbol, Defined Symbol, or Integer Number
-9	Any Keyword
-10	Symbol or Keyword
-11	List of Symbols
-12	List of Symbols, Defined Symbols, or Literals
-15	Hollerith String

c. Blank Common /BLNKCM/

```
LEVEL 2, BLNKCM
COMMON BLNKCM(3000)
DIMENSION IBLKCM(3000)
EQUIVALENCE (BLNKCM(1), IBLKCM(1))
```

The purpose of common block BLNKCM is to store data for manipulation within EXEMPT. The size of blank common is dynamically adjusted as required to accommodate the data.

d. Common /CHAR/

```
COMMON/CHAR/IALPH(26),IDIG(10),ISYM(6),MONOPR(6),MAXALP,MAXNUM,
, MAXSYM,MAXOPR,LOCPLS,LOCMIN,LOCMUL,LOCDIV,LOCEQU,
, LOCEXP,MAXINT,NUMDEC,MAXFPT,MAXPWR,LOCDOl,LOCITP,
, LOCRTP,LOC COM,LOC PER,LOC BLK
```

Common block CHAR contains all of the legal characters within EXEMPT.

Table 12

/CHAR/

<u>Name</u>	<u>Function</u>
IALPH	Contains all the alphabetic characters A - Z
IDIG	Contains all the numeric characters 1 - 9
ISYM	Contains all the symbols, \$, (,), , , and blank
LOCBLK	Contains the location of the blank in ISYM
LOCCOM	Contains the location of the comma in ISYM
LOCDIV	Contains the location of the division sign in MONOPR
LOCDOL	Contains the location of the dollar sign in ISYM
LOCEQU	Contains the location of the equal sign in MONOPR
LOCEXP	Contains the location of the exponential sign in MONOPR
LOCLTP	Contains the location of the left parenthesis in ISYM
LOCMIN	Contains the location of the minus sign in MONOPR
LOCMUL	Contains the location of the multiply sign in MONPR
LOCPER	Contains the location of the period or decimal in ISYM
LOCPLS	Contains the location of the plus sign in MONOPR
LOCRTP	Contains the location of the right parenthesis in ISYM

Table 12
/CHAR/ (Concluded)

<u>Name</u>	<u>Function</u>
MAXALP	Contains the maximum number of alphabetic characters
MAXFPT	Contains the maximum number of floating point digits
MAXINT	Contains the maximum number of integer digits
MAXNUM	Contains the maximum number of digits in the table IDIG
MAXOPR	Contains the maximum number of operators in the table MONOPR
MAXPWR	Contains the maximum number of digits allowed after the E in E format
MAXSYM	Contains the maximum number of symbols contained in the table ISYM
MONOPR	This is the table that contains all of the allowable operators within EXEMPT (+, -, *, 1, =, **)
NUMDEC	This contains the maximum number of digits allowed after the decimal point

e. Common /DEFLST/

```
COMMON/DEFLST/FRQFIL,FRQMAX,FRQMIN,GREALK20),NUMBCM,PAGLEN,PAGWID,
      SPCM,TIMFIL,TIMMAX,TIMMIN,VPCM,ZERO
      INTEGER FRQFIL,TIMFIL,PAGLEN,PAGWID
```

Common block DEFLST contains all default valued keywords with their initial values. These keywords may have their values changed during execution and may be used as operands in arithmetic statements.

Table 13

/DEFLST/

<u>Name</u>	<u>Function</u>
FRQFIL	Contains the sequential file that has been designated to contain frequency only data
FRQMAX	Contains the maximum frequency value
FRQMIN	Contains the minimum frequency value
GREAL	The global real array
NUMBCM	Contains the number of blank common locations available
PAGLEN	Contains the length of the page to be printed on
PAGWID	Contains the width of the page to be printed on
SPCM	Not used
TIMFIL	Contains the sequential file that was designated to contain time only data
TIMMAX	Contains the maximum time value
TIMMIN	Contains the minimum time value
VPCM	Not used
ZERO	Used for closeness testing for independent values - value is defaulted to 1.E-10

f. Common /DSKNAM/

```
COMMON /DSKNAM/ IFLNOW,IBLKNM(20),IFILID(2715),IDD(5),LISTDM,  
,PRINT,LISTRE  
LOGICAL LISTDM,LISTRE,PRINT
```

Common block DSKNAM contains all information necessary in communicating with the random access system.

Table 14
/DSKNAM/

<u>Name</u>	<u>Function</u>
IBLKNM	Contains all of the file names within the random access system
IDD	Contains the current index to the current file being used
IFILID	Contains the directory of the current file on the random access system in use
IFLNOW	Contains the current file in use
LISTDM	Not used
LISTRE	Not used
PRINT	Not used

g. Common /ERMODE/

```
COMMON/ERMODE/ICOND,IERROR,IWARN,IDANGR,IERCOD,MAXSTK,IDECEL,  
, MAXSTR,MAXCHR,MXINCT,MXFPC,NFIELD,ILLCHR,MXPCT,  
, KWERR
```

Common block ERMODE contains all of the error codes detected in the subroutine SCAN of the LIP.

Table 15

/ERMODE/

<u>Name</u>	<u>Function</u>
ICOND	Contains the condition of the type of error
IDANGR	Is a code for ICOND indicating the scanning tables have overflowed
IDECER	Indicates that a decimal was illegally used
IERCOD	Contains the type of error that occurred
IERROR	Indicates an error mode for ICOND
ILLCHR	Indicates that an illegal character was used
IWARN	Indicates a warning message is to be issued and the condition for ICOND
KWERR	Indicates that a keyword error has occurred
MAXCHR	Indicates that the maximum number of characters for a symbol has been exceeded
MAXSTK	Indicates that the maximum stack length has been exceeded for the scan tables
MAXSTR	Indicates that the maximum Hollerith string length has been exceeded
MXFPCT	Indicates that the maximum floating point length has been exceeded
MXINCT	Indicates that the maximum integer length has been exceeded
MXPRCT	Indicates that the power of the floating point is too large
NFIELD	Not used

c. Common /ERR/

COMMON /ERR/ RDERR,NE, IDUMP, IDPLT
LOGICAL RDERR

Common block ERR indicates that an error has occurred on a random access file.

Table 16

/ERR/

<u>Name</u>	<u>Function</u>
IDPLT	Not used
IDUMP	Indicates the number of words to be dumped around the word, a fatal error that occurred within EXEMPT
NE	Not used
RDERR	Indicates that a read error will occur if the user attempts to read a random access file that has not yet been opened

i. Common /FITS/

COMMON/FITS/ NFWOBUF,NFWBUF,IFITWO(20),IFITW(5)

Common block FITS is used to indicate the current sequential files being used. There are 20 possible sequential files that may be user designated. Ten will have buffers, and ten will be without buffers.

Table 17
/FITS/

<u>Name</u>	<u>Function</u>
IFITW	Not used
IFTWO	Contains all of the sequential files that have been user designated
NFWBUF	Contains the number of files on the program card with buffers of the form BTBUFX
NFWOBUF	Contains the number of files on the program card that do not contain buffers of the form BTIOXX

j. Common /HOLSTR/

COMMON/HOLSTR/LHSTRG(25,12),MXHSTR,MAXHSL,NUMCHR,LOCHS,LTITLE

Common block HOLSTR is the storage area for the Hollerith string. The argument list table will contain a pointer for a Hollerith string to the array LHSTRG.

Table 18

/HOLSTR/

<u>Name</u>	<u>Function</u>
LHSTRG	Contains all of the Hollerith string or titles that were created within a run stream
LOCHS	Contains the next available location within LHSTRG for a Hollerith string
LTITLE	This is the flag to indicate that a Hollerith string is being created
MAXHSL	Contains the maximum Hollerith string length of 120 characters
MXHSTR	Contains the maximum number of Hollerith strings within a run stream -- maximum number is 25
NUMCHR	When a Hollerith string is being scanned, this contains the current character count

k. Common /KEYWRD/

COMMON/KEYWRD/KWNAME(256*2),MAXKW

Common block KEYWRD contains all the keywords with EXEMPT.

Table 19

/KEYWRD/

<u>Name</u>	<u>Function</u>
KWNAME	Is the array that contains all keywords
MAXKW	Contains the maximum number of keywords within EXEMPT (256)

1. Common /KWNAM1/

COMMON/KWNAM1/KWALFL, KWCLPT, KWCALL, KWDROP, KWEOF, KWFILE,
A KWFMPY, KWGETR, KWGDPT, KWLST, KWFORM, KWPRT, KWPUCH,
B KWREAD, KWREAC, KWREWD, KWRUN, KWSAVE, KWSCAN, KWSKPB,
C KWSKPF, KWSAVR, KWTITE, KWWRITE, KWRNAM, KWREDF, KWLOOP,
D KWLBEL, KWCHGE, KWEND, KWPURG, KWTYPE, KWFUN, KWBUG

Common block KWNAM1 contains all the indices to the system commands within EXEMPT.

Table 20

/KWNAM1/

<u>Name</u>	<u>Function</u>
KWALFL	Contains the index to the system command for the keyword ALTFIL
KWCALL	Contains the index to the keyword table for the system command CALL
KWCHGE	Not used
KWREAD	Contains the index to the keyword table for the system command READ
KWREDF	Contains the index to the keyword table for system command READF
KWREWD	Contains the index to the keyword table for the system command REWIND

Table 20
/KWNAM1/ (Concluded)

<u>Name</u>	<u>Function</u>
KWRITE	Contains the index to the keyword table for the system command WRITE
KWRNAM	Contains the index to the keyword table for the system command RENAME
KWRUN	Contains the index to the keyword table for the system command RUN
KWSAVE	Contains the index to the keyword table for the system command SAVE
KWSAVR	Contains the index to the keyword table for the system command SAVRAN
KWSCAN	Not used
KWSKPB	Contains the index to the keyword table for the system command SKIPB
KWSKPF	Contains the index to the keyword table for the system command SKIPF
KWTITE	Contains the index to the keyword table for the system command title
KWTYPE	Contains the index to the keyword table for the system command TYPE

m. Common /KWNAM2/

COMMON/KWNAM2/KWABS, KWATAN, KWATN2, KWCLEN, KWCONJ, KWCNVT,
A KWCOS, KWEXP, KWFT, KWCMPX, KWINTP, KWINVT, KWALOG,
B KWMAX, KWMEGE, KWMIN, KWMOD, KWEDIT, KWSHFT, KWSIGN,
C KWSIN, KWSQRT, KWTAN, KWDM9, KWTDEP, KWTDSN, KWW,
D KWDM9, KWD10, KWD11

Common block KWNAM2 contains the indices to the keyword table for
all keywords that are used as functions.

Table 21

/KWNAM2/

<u>Name</u>	<u>Function</u>
KWABS	Contains the index to the keyword table for the function ABS
KWALOG	Contains the index to the keyword table for the function ALOG
KWATAN	Contains the index to the keyword table for the function ATAN
KWATN2	Contains the index to the keyword table for the function ATAN2
KWCMPX	Contains the index to the keyword table for the function CMPLX
KWCNVT	Contains the index to the keyword table for the function CONVRT
KWCONJ	Contains the index to the keyword table for the function CONJG
KWCOS	Contains the index to the keyword table for the function COS
KWEDIT	Contains the index to the keyword table for the function EDIT
KWEXP	Contains the index to the keyword table for the function EXP
KWFT	Contains the index to the keyword table for the function FT

Table 21
/KWNAM2/ (Continued)

<u>Name</u>	<u>Function</u>
KWINTP	Contains the index to the keyword table for the function INTERP
KWINVT	Contains the index to the keyword table for the function INVFT
KWMAX	Contains the index to the keyword table for the function MAX
KWMEGE	Contains the index to the keyword table for the function MERGE
KWMIN	Contains the index to the keyword table for the function MIN
KWMOD	Contains the index to the keyword table for the function MOD
KWPOLY	Contains the index to the keyword table for the function WPOLY
KWSHFT	Contains the index to the keyword table for the function SHIFT
KWSIGN	Contains the index to the keyword table for the function SIGN
KWSIN	Contains the index to the keyword table for the function SIN
KWSQRT	Contains the index to the keyword table for the function SQRT
KWTAN	Contains the index to the keyword table for the function TAN
KWTDEP	Contains the index to the keyword table for the function TRDEXP
KWTDSN	Contains the index to the keyword table for the function TRDSIN

n. Common /KWNAM3/

COMMON/KWNAM3/KWAVBA,KWCOCK,KWCOND,KWENSI,KWHFAN,KWINDC,KWMARB,
A KWPITO,KWRDAL,KWRDAH,KWRVBE,KWSHIE,KWWEAB,KWWEBA,KWWR00,
B KWWWEL,KWWNGN,KWSNGL,KWAS19,KWCOAX,KWSFLD,KWAVGS,KWCOUP,
C KWPNLJ,KWDBCB,KWDBC1

Common block KWNAM3 contains the indices to the keyword table for the SLAM keywords.

Table 22

/KWNAM3/

<u>Name</u>	<u>Function</u>
KWAS19	Contains the index to the keyword table for the SLAM function AS1918
KWAVBA	Contains the index to the keyword table for the SLAM function AVBAY
KWAVGS	Contains the index to the keyword table for the SLAM function AVGSKT
KWCOAX	Contains the index to the keyword table for the SLAM function COAX
KWCOCK	Contains the index for the keyword table for the SLAM function COCKPT
KWCOND	Contains the index to the keyword table for the SLAM function CONDUIT
KWCOUP	Contains the index to the keyword table for the SLAM function COUPLER
KWDBCB	Contains the index to the keyword table for the SLAM function DBCAB
KWDBC1	Contains the index to the keyword table for the SLAM function DBCABI
KWENGI	Contains the index to the keyword table for the SLAM function ENGINE
KWHFAN	Contains the index to the keyword table for the SLAM function HFANT

Table 22
/KWNAM3/ (Concluded)

<u>Name</u>	<u>Function</u>
KWINDC	Contains the index to the keyword table for the SLAM function INDCOU
KWMARB	Contains the index to the keyword table for the SLAM function MARBEA
KWPITO	Contains the index to the keyword table for the SLAM function PITOT
KWPNLJ	Contains the index to the keyword table for the SLAM function PN LJNT
KWRDAL	Contains the index to the keyword table for the SLAM function RADALT
KWRDAR	Contains the index to the keyword table for the SLAM function RADAR
KWRVBE	Contains the index to the keyword table for the SLAM function RVBEA
KWSFLD	Contains the index to the keyword table for the SLAM function SFIELD
KWSHIE	Contains the index to the keyword table for the SLAM function SHIELD
KWSNGL	Contains the index to the keyword table for the SLAM function SNGLIN
KWEAAB	Contains the index to the keyword table for the SLAM function WEABAY
KWEABA	Contains the index to the keyword table for the SLAM function WEBAOP
KWWROO	Contains the index to the keyword table for the SLAM function WROOT
KWWWEL	Contains the index to the keyword table for the SLAM function WWELL
KWWWCN	Contains the index to the keyword table for the SLAM function WWGND

o. Common /KWNAM4/

COMMON/KWNAM4/KWDBLN, KWDBLG, KWINDP, KWDGRS, KWDCFL, KWCURF,
 A KWNDEX, KWLOG, KWLIN, KWLNLN, KWLNLG, KWLGLN,
 B KWLGGL, KWMXVL, KWMNVL, KWNPTS, KWMP, KWOFF, KWON,
 C KWPART, KWPT1, KWPT2, KWRADS, KWR1, KWRDB,
 D KWSAFE, KWSVFN, KWSGLE, KWTLEX, KWTLEY, KWREP, KWNXT,
 E KWDLFQ, KWDLTM, KWFQMX, KWFQMN, KWSPCM, KWTMMX, KWTMMN,
 F KWVPCM, KWZERO, KWFQFL, KWTMFL,
 G KWR1, KWR2, KWR3, KWR4, KWR5, KWR6, KWR7, KWR8, KWR9,
 H KWR10, KWR11, KWR12, KWR13, KWR14, KWR15, KWR16, KWR17,
 I KWR18, KWR19, KWR20, KWPAGL, KWPAGW

Common block KWNAM4 contains the default value and parameter keyword indices to the keyword table.

Table 23

/KWNAM4/

<u>Name</u>	<u>Function</u>
KWCURF	Contains the pointer to the keyword table for the keyword CURFIL
KWDB	Contains the pointer to the keyword table for the keyword DB
KWDBLG	Contains the pointer to the keyword table for the keyword DBLOG
KWDBLN	Contains the pointer to the keyword table for the keyword DBLIN
KWDCFL	Contains the pointer to the keyword table for the keyword DOCFIL
KWDGRS	Contains the pointer to the keyword table for the keyword DGREES
KWDLFQ	Contains the pointer to the keyword table for the keyword DELFRQ
KWDLTM	Contains the pointer to the keyword table for the keyword DELTIN
KWFQFL	Contains the pointer to the keyword table for the keyword FRQFIL
KWFQMN	Contains the pointer to the keyword table for the keyword FRQMIN
KWFQMX	Contains the pointer to the keyword table for the keyword FRQMAX

Table 23

/KWNAM4/ (Continued)

<u>Name</u>	<u>Function</u>
KWINDP	Contains the pointer to the keyword table for the keyword INDPNT
KWLBNM	Not Used
KWLGLG	Contains the pointer to the keyword table for the keyword LOGLOG
KWLGLN	Contains the pointer to the keyword table for the keyword LOGLIN
KWLIN	Contains the pointer to the keyword table for the keyword LIN
KWLNLG	Contains the pointer to the keyword table for the keyword LINLOG
KWLNLN	Contains the pointer to the keyword table for the keyword LINLIN
KWLOG	Contains the pointer to the keyword table for the keyword LOG
KWMNVL	Contains the pointer to the keyword table for the keyword MINVAL
KWMP	Contains the pointer to the keyword table for the keyword MP
KWMXVL	Contains the pointer to the keyword table for the keyword MAXBAL
KWNDEX	Contains the pointer to the keyword table for the keyword INDEX
KWNPTS	Contains the pointer to the keyword table for the keyword MPTS
KWNXT	Contains the pointer to the keyword table for the keyword NXT
KWOFF	Contains the pointer to the keyword table for the keyword OFF
KWON	Contains the pointer to the keyword table for the keyword ON

Table 23
 /KWNAM4/ (Continued)

<u>Name</u>	<u>Function</u>
KWPAGL	Contains the pointer to the keyword table for the keyword PAGLEN
KWPAGW	Contains the pointer to the keyword table for the keyword PAGWID
KWPART	Contains the pointer to the keyword table for the keyword PART
KWPT1	Contains the pointer to the keyword table for the keyword PT1
KWPT2	Contains the pointer to the keyword table for the keyword PT2
KWRADS	Contains the pointer to the keyword table for the keyword RADS
KWREP	Contains the pointer to the keyword table for the keyword REP
KWRI	Contains the pointer to the keyword table for the keyword RI
KWR1-KWR20	Contains the indexes to the keyword table for the keywords R1-R20
KWSAFE	Contains the index to the keyword table for the keyword SAFE
KWSGLE	Contains the pointer to the keyword table for the keyword SINGLE
KWSPCM	Contains the pointer to the keyword table for the keyword SPCM
KWSVFN	Contains the pointer to the keyword table for the keyword SAVEFN
KWTLEX	Contains the pointer to the keyword table for the keyword TITLEX
KWTLEY	Contains the pointer to the keyword table for the keyword TITLEY
KWTMFL	Contains the pointer to the keyword table for the keyword TIMFIL
KWTMMN	Contains the pointer to the keyword table for the keyword TIMMIN

Table 23

/KWNAM4/ (Concluded)

<u>Name</u>	<u>Function</u>
KWTMMX	Contains the pointer to the keyword table for the keyword TIMMAX
KWVPCM	Contains the pointer to the keyword table for the keyword VPCN

p. Common /KWNAM5/

COMMON/KWNAM5/KWAEFF,KWALPH,KWC0 ,KWC1,KWC2,KWC3,KWD1,KWD2,KWD3,
 A KWDIV,KWEPS,KWFLTF,KWFMAX,KWFMIN,KWHEIG,KWL1,KWL2,KWL3,
 B KWL4,KWLENG,KWM1,KWMODE,KWPHI,KWTAU,KWTHT,KWWIDT,KWZ0,
 C KWZ0,KWZS,KWZT,KWANGL,KWWEIG,KWF1,KWWIRE,KWDELT,KWDEPT,
 D KWSIGM,KWA,KWB,KWC,KWN1,KWN2,KWA1,KWA2

Common block KWNAM5 contains the keywords used as parameters for the SLAM functions.

Table 24

/KWNAM5/

<u>Name</u>	<u>Function</u>
KWA	Contains a pointer to the keyword table for the keyword A
KWAEFF	Contains the pointer to the keyword table for the keyword AEFF
KWALPH	Contains the pointer to the keyword table for the keyword ALPH
KWANGL	Contains the pointer to the keyword table for the keyword ANGLE
KWA1-KWA2	Contains the pointers to the keyword table for the keywords A1, A2
KWB	Contains the pointer to the keyword table for the keyword B
KWC	Contains the pointer to the keyword table for the keyword C
KWCO-KWC3	Contains the pointers to the keyword table for the keywords CO - C3
KWDELT	Contains the pointers to the keyword table for the keyword DELTA

Table 24

/KWNAM5/ (Continued)

<u>Name</u>	<u>Function</u>
KWDEPT	Contains the pointer to the keyword table for the keyword DEPTH
KWDIV	Contains the pointer to the keyword table for the keyword DIV
KWD1-KWD3	Contains the pointer to the keyword table for the keywords D1 - D3
KWEPS	Contains the pointer to the keyword table for the keyword EPS
KWFLTF	Contains the pointer to the keyword table for the keyword FLTFRQ
KWFMAX	Contains the pointer to the keyword table for the keyword FMAX
KWFMIN	Contains the pointer to the keyword table for the keyword FMIN
KWF1	Contains the pointer to the keyword table for the keyword F1
KWHEIG	Contains the pointer to the keyword table for the keyword HEIGHT
KWLENG	Contains the pointer to the keyword table for the keyword LENGTH
KWL1-KWL4	Contains the pointer to the keyword table for the keywords L1 - L4
KWMODE	Contains the pointer to the keyword table for the keyword MODE
KWM1	Contains the pointer to the keyword table for the keyword M1
KWN1-KWN2	Contains the pointer to the keyword table for the keywords N1, N2.
KWPHI	Contains the pointer to the keyword table for the keyword PHI

Table 24
 /KWNAM5/ (Concluded)

<u>Name</u>	<u>Function</u>
KWSIGM	Contains the pointer to the keyword table for the keyword SIGMA
KWTAU	Contains the pointer to the keyword table for the keyword TAU
KWTHET	Contains the pointer to the keyword table for the keyword THETA
KWWEIG	Contains the pointer to the keyword table for the keyword WEIGHT
KWWIDT	Contains the pointer to the keyword table for the keyword WIDTH
KWWIRE	Contains the pointer to the keyword table for the keyword WIRE
KWZ0 and KWZØ	Contains the pointers to the keyword table for the keywords Z0 or ZØ
KWZS	Contains the pointer to the keyword table for the keyword ZS
KWZT	Contains the pointer to the keyword table for the keyword ZT

q. Common /KWTASK/

COMMON/KWTASK/KWTSTB(400)

Common block KWTASK contains the task tables for each system, arithmetic, and SLAM function.

Each function is parsed from the information given in the NCODE and NVAL tables according to the rigid structure in the keyword task table (KWTSTB). For example, the TITLE command structure is

TITLE DS, FILE=SDSN, HS

where

DS = Defined Symbol
SDSN = Symbol, Defined Symbol, or Number
HS = Hollerith String

When the TITLE command is parsed, it is placed in the argument list table as shown in Table 25.

Table 25
ARGUMENT LIST TABLE*

	CODE	DEFINITION
.	.	.
.	.	.
.	.	.
DS	0<CODE<1000	Pointer to the symbol table for the symbol to be titled
SDSN	CODE<0, #-999999	Absolute value points to the literal table for the alpha or numeric value
HS	CODE<0, #-999999	Absolute value points to the literal table for the pointer to the Hollerith string table
TITLE	1030	Biased code for the keyword TITLE
EOS	1007	End of statement code
.	.	.
.	.	.
.	.	.
.	.	.

* Reference Section I-B for a description of how the argument table is built.

If a no-op code (-999999) is present in the argument, then the parameter for that position was not given; however, it is up to the execution phase to determine whether or not that parameter is required or optional.

r. Common /LOCDAT/

COMMON /LOCDAT/ NUMOPS, LOCD(4,2), ITYPED(4), NUMD(4), LNK(4),
1 MAGRI(4)

Common block LOCDAT maintains the current linkage to the data set entries in blank common.

Table 26

/LOCDAT/

<u>Name</u>	<u>Function</u>
ITYPED	Contains the current type of data being requested or stored on random access. A positive number indicates frequency data, a negative number indicates time data, and a zero indicates that the data is an independent or a no-type data set
LINK	Contains the pointer to the symbol table for the data set being currently stored or fetched
LOCD	Contains the pointer to the starting address for the data set in the array BLNCKM
MAGRI	Contains the attribute of the data set being requested or stored on random access
NUMD	Contains the number of points contained in the data sets being stored or requested from random access
NUMCPS	Contains the number of data sets that are being manipulated in the array BLNCKM in an arithmetic operation

s. Common /MSTRG/

COMMON /MSTRG/ MSTNDX(22), IUNDX(56), IUN(20), NUND, GETDATA, PUTDATA,
, DFET(25), LISTDSK
LOGICAL LISTDSK, GETDATA, PUTDATA

Common block MSTRG contains all information needed for accessing the random access system at the master index level.

Table 27

/MSTRG/

<u>Name</u>	<u>Function</u>
DFET	Contains the pointer to the name table for the current file on random access being used
IUN	Contains the table of all the files that are active on the random access
IUNDX	Contains the directory of all of the subelements on the master index
MSTNDX	Contains the master index of the random access file on unit 99
NUND	Contains the current number of active files on random access

t. Common/PLTFMT/

COMMON/PLTFMT/ SPX, SPY, AXISX, AXISY, PAGEX, PAGEY

Common block PLTFMT contains the plot format data.

All data is expressed centimeters.

Table 28

/PLTFMT/

<u>Name</u>	<u>Function</u>
AXISX	Length of the X axis
AXISY	Length of the Y axis
PAGEX	Length of plot page along X axis
PAGEY	Length of plot page along Y axis
SPX, SPY	X, Y coordinates on the plot page for the intersection of the X and Y axis (Lower left corner of the plot frame)

u. Common /PLTLAB/

COMMON/PLTLAB/ LABX(2), LABY(2), LAB(34), NXFMT, NYFMT, PLTERR
LOGICAL PLTERR

Common PLTLAB contains Hollerith variables used to specify plot axis and frame labels, and plot axis annotation formats.

Table 29

/PLTLAB/

<u>Name</u>	<u>Function</u>
LAB	Used for annotating the plot
LABX	Contains the X axis label
LABY	Contains the Y axis label
NXFMT, NYFMT	Hollerith variable used to specify the format of the X and Y axis number annotations. Formats are G, E, F, and I. Default is NXFMT = 7H1PG12.2
PLTERR	Logical variable used to indicate a plot package error. .TRUE. indicates error

v. Common /PLTLGV/

COMMON/PLTLGV/ PLTLBL, FULFRM, NOZL, LABVRT, NX10PW, LABPAG,
1 LCUTMK, LSQNUM, STPFLG, PLTING, PLTOPN, PLTCLO
LOGICAL PLTLBL, FULFRM, NOZL, LABVRT, NX10PW, LABPAG
LOGICAL LCUTMK, LSQNUM, STPFLG, PLTING, PLTOPN, PLTCLO

Common block PLTLGV contains all logical variables used in the plot package routines.

Table 30

/PLTLGV/

<u>Name</u>	<u>Value</u>	<u>Function</u>
FULFRM	.TRUE.	Controls the generation of the plot frame. When .TRUE. a full plot frame, when .FALSE. only X and Y axis

Table 30

<u>Name</u>	<u>Value</u>	<u>Function</u>
		<i>/PLTLGV/ (Continued)</i>
LABPAG	.TRUE.	Controls plot page annotation
LABVRT	.FALSE.	Specifies orientation of the Y axis label. When .TRUE. a vertical Y axis, when .FALSE. a label is parallel to the Y axis
LCUTMK	.TRUE.	Controls generation of plot cut mark guides
LSQNUM	.FALSE.	Controls automatic plot sequence number annotation
NOZL	.FALSE.	Controls automatic generation of a zero reference line on plots. When .TRUE. no zero reference lines will be drawn. When .FALSE. a zero reference line will appear only when any Y values are less than zero
NX10PW	.FALSE.	Controls exponential format annotation
PLTCLO	.FALSE.	Indicates the plot file has been closed. Set .TRUE. when CLOSEPP is called
PLTING	.FALSE.	Indicates a plot is in progress. If .TRUE. then implies that a plot frame has been drawn and plot limits have been defined
PLTLBL	.TRUE.	Used to inhibit plot labels. A .TRUE. causes plots to be labeled.
PLTOPN	.FALSE.	When .TRUE. indicates a plot file has been opened. Set .TRUE. when OPENPP is called
STPFLG	.TRUE.	Indicates how plot package is to handle error situations

w. Common /PLTCM1/, /PLTCM2/, /PLTCM3/, and /PLTCM4/

COMMON/PLTCM1/ TML, HNO, HLB, HML, MDECX, MDECY, MXI, MYI, CALCP,
1 PID, PDAT, PTIM

COMMON/PLTCM2/ CW, LOGERR, IPGNUM, LUNMPF, LUNPIF, IBLNK, IMESG,
1 XPSF, YPSF

COMMON/PLTCM3/ XP, YP, HL, IL, SINA, COSA, ICHAR(20)

COMMON/PLTCM4/ XMIN, XMAX, YMIN, YMAX, XS, YS, IDPCOM, SFLX, SFNX

These common blocks are used to support the plot package when
making a METALIB plot file.

Table 31

/PLTCM1/

<u>Name</u>	<u>Function</u>
CALCP	Resolution of CALCOMP plotter in steps/ inch (Default is 100)
HLB	Height of plot labels in cm
HML	Height of message label characters in cm
HNO	Height of axis annotation numbers in cm
MDECX	Maximum number of decades for a logarithmic X axis (Default is 6)
MDECY	Maximum number of decades for a logarithmic Y axis (Default is 6)
MXI	Maximum number of intervals along the X axis (Default is 10)
MYI	Maximum number of intervals along the Y axis (Default is 10)
PDAT	Hollerith variable containing the system data
PID	Hollerith variable containing the seven character job ID
PTIM	Hollerith variable containing the system time
TML	Length of plot tic marks in cm

Table 32

/PLTCM2/

<u>Name</u>	<u>Function</u>
CW	Label character width to height ratio (6./7.)
IBLNK	Hollerith variable containing 10 blanks
IMESG	Counter for the number of messages per plot
IPGNUM	Sequential plot page number
LOGERR	Hollerith variable used to specify the method for handling number .LE. 0 when plotted on a logarithmic axis. This variable can be user specified by calling LSERR
LUNMPF	Contains the logical unit number for a METAPLOT file
LUNPIF	Contains the logical unit number for plot information file
XPSF, YPSF	X and Y plot array scale factors. (Scale factors must be greater than zero. Default is 1.) Scale factors may be set by the user through a call to RSCALE. Scale factors remain in effect until another call to RSCALE is made

Table 33

/PLTCM3/

<u>Name</u>	<u>Function</u>
COSA	Cosine of the axis angle
SINA	Sine of the axis angle
	NOTE - X axis angle = 0 degrees Y axis angle = 90 degrees
HL	Temporary character height
ICHAR	Array used for character manipulation

Table 33
/PLTCM3 (Concluded)

<u>Name</u>	<u>Function</u>
IL	Temporary integer variable
XP, YP	Temporary X and Y pin positions

Table 34
/PLTCM4/

<u>Name</u>	<u>Function</u>
IDPCOM	Temporary variable for user-supplied plot ID number
SFLX	Scale factor for X axis label character size
SFNX	Scale factor for X axis number character size
XMAX	Maximum plot limit on X axis
XMIN	Minimum plot limit on X axis
XS	Distance in X data coordinates for tic mark length
YMAX	Maximum plot limit on Y axis
YMIN	Minimum plot limit on Y axis
YS	Same as XS but for Y axis

x. Common /TMPFIL/

COMMON /TMPFIL/ JFET(2, 20), JUNIT(20), KURSQF(20), JNUM, JFILE

The purpose of this common block is to maintain the file environment tables for the sequential files and to contain the list of active sequential files within EXEMPT.

Table 35

/TMPFIL/

<u>Name</u>	<u>Function</u>
JFET	Contains the file environment table for all active sequential files
JFILE	Contains the pointer to the file environment table for the current active sequential file
JNUM	Contains the number of active sequential files
JUNIT	Contains the table of names of all active sequential files
KURSQF	Contains the table that keeps track of which EOF mark each sequential file is at

y. Common /XQTTBL/

```
COMMON/XQTTBL/ NEQUAL, IEOS, LSTARG(256), MAXVAL, NOGOFG, NUMARG,
          IPASS, KWFNST, KWFNND, ZDEBUG, ZRAND
LOGICAL ZDEBUG, ZRAND, NOGOFG
```

Common block XQTTBL stores temporary information for the execution of the user's input.

Table 36

/XQTTBL/

<u>Name</u>	<u>Function</u>
IEOS	Contains the code for an end of statement
IPASS	Contains the execution pass number
KWFNND	Contains the last system function
KWFNST	Contains the first system function
LSTARG	Contains the argument list for one statement

Table 36
/XQTTBL/ (Concluded)

<u>Name</u>	<u>Function</u>
MAXVAL	Contains the maximum number of fields in the table LSTARG
NEQUAL	Contains the code for an equal sign
NOGOFG	Logical flag that when true means an error has occurred in execution
NUMARG	The current number of arguments within the table LSTARG
ZDEBUG	Not Used
ZRAND	Not Used

SECTION III

FUNCTIONAL FLOWCHARTS

1. LOGICAL OPERATIONS

The CLP is the main processor controlling the execution of the logical operations. The commands are either processed and executed within the CLP or the CLP passes control to other routines. The logical operations are grouped into the categories of sequential I/O, random I/O, and coded I/O.

a. Sequential I/O Function

The structure of the sequential file is in terms of data set. Motion of the sequential file is done by sets rather than records. A READ will read one data set and a WRITE will write one data set.

The first record of a set will contain the number of points, data description, date, program name, and the title information. A time domain data type will have two other records and a frequency domain data type will contain three other records. The structure of these data types are:

a. Frequency Domain Data (Four Records)

1. Record 1: 17 words long
 - (a) One word integer (N) indicating number of words in each of the following records.
 - (b) One word containing data description in CDC display code.
 - (c) One word containing data description of data values are 8HMAGPHASE, 8HREALIMAG, 7HDBPHASE.
 - (d) One word containing a data in display code.
 - (e) One word containing display code or name of program which wrote data file (e.g., 7HSCEPTRE, 6HSYNAP, etc.)
 - (f) Set attributes.
 - (g) 12 words of title information in CDC display code.
2. Record 2: N words long containing the array of frequency points.
3. Record 3: N words long containing the real or amplitude of a frequency function.
4. Record 4: N words long containing the imaginary part or phase of a frequency function.

b. Time Domain Data (Three Records)

1. Record 1: 17 words long; same as for frequency data.
2. Record 2: N words long containing the array of time point.
3. Record 3: N words long containing the values of the time function.

The sequential file I/O functions are preprocessed within SEQCMD prior to calling the sequential file driving routine EXINTO. This routine is responsible for all sequential file manipulations in the format described above. Figures 11 to 15 are functional flowcharts of all the sequential file I/O operations available within EXEMPT.

b. Random I/O Functions

The random I/O functions determine the type of the data, the format of the data, how the data are stored, and other information required by the user. Again, these functions are controlled by the CLP which calls the routines necessary for executing the random I/O functions. The flowcharts for these functions are shown in Figures 16 to 23.

c. Coded I/O Functions

Provided within these functions are the plotting, printing, punching, and card reading functions for the user. These functions are illustrated in Figures 24 to 27.

2. SLAM FUNCTIONS

When a SLAM function is detected in the CLP, control is immediately passed to the SLAM function processor (SUBDRV) for processing. The function of this processor is illustrated in Figure 28. The first operation is to initialize the global real array to zeros. The global real array is used to pass all the optional parameters. If a parameter is missing, then the entry in the global real array for that parameter remains zero. When a SLAM function detects a zero entry, a default value is assigned to that parameter.

After the input and output data set names have been determined, then control is passed by a computed GOTO to the appropriate section of code in SUBDRV for preprocessing of the SLAM function. This entails fetching a global independent (TIME or FREQ), error checking, and fetching the input data sets. The SLAM function is then called with the global real array, independent array, and the specified input and output data sets. When the

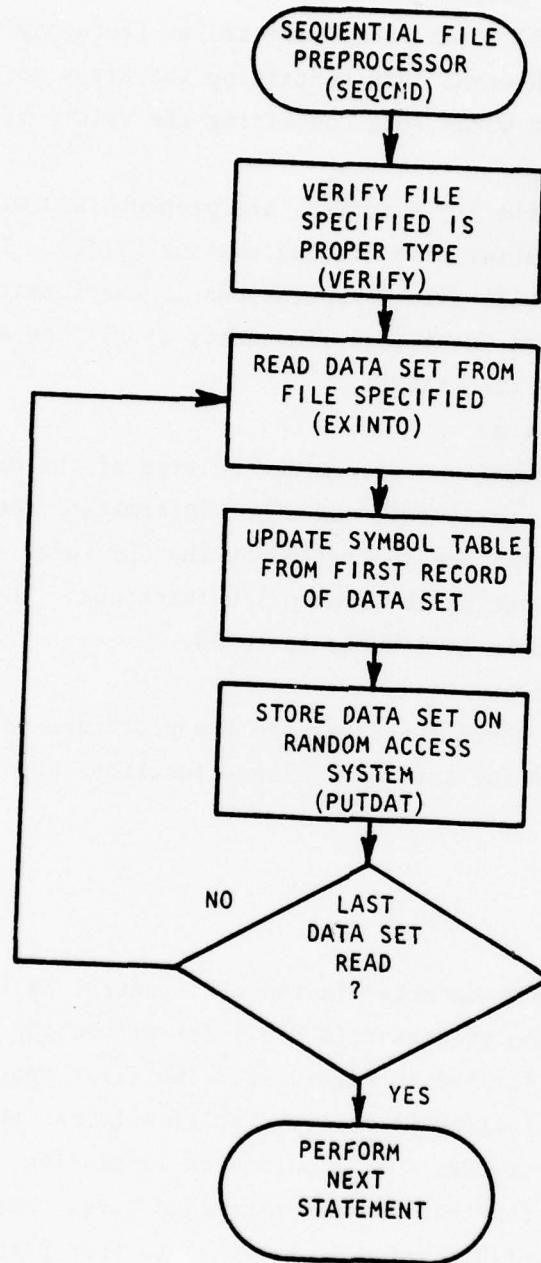


Figure 11. READ Function Flowchart

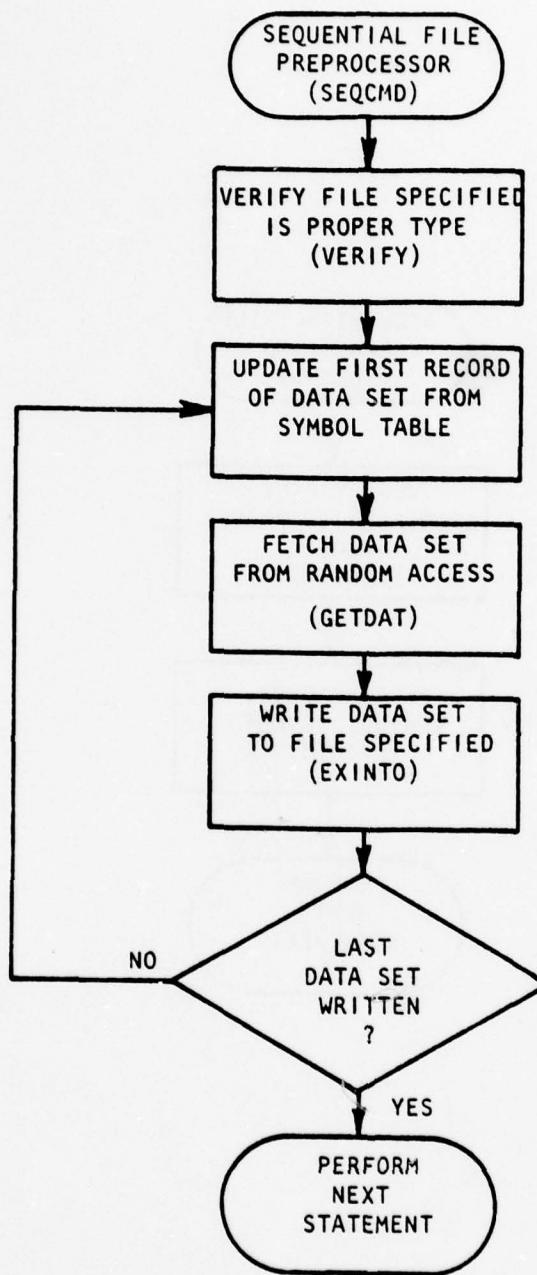


Figure 12. WRITE Function Flowchart

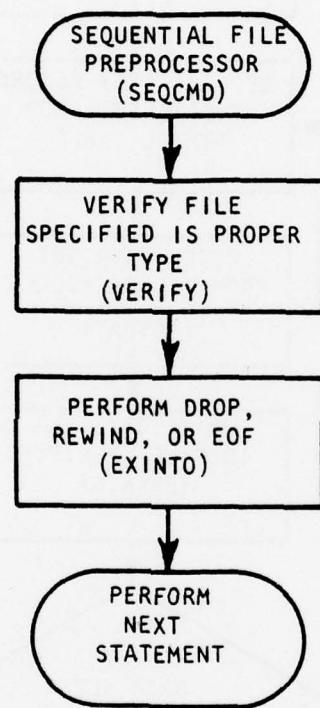


Figure 13. DROP, EOF, and REWIND Function Flowchart

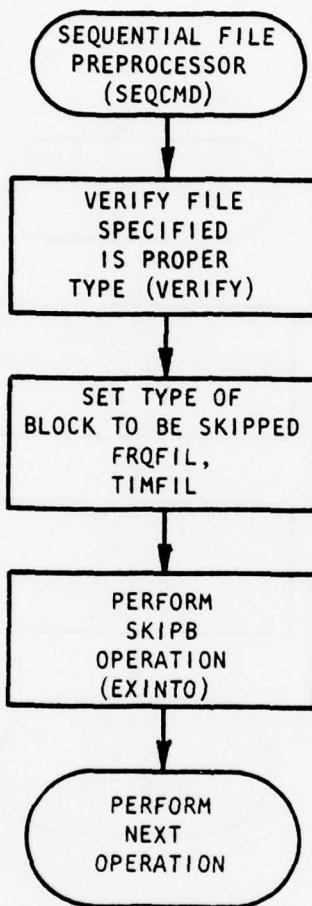


Figure 14. SKIPB Function Flowchart

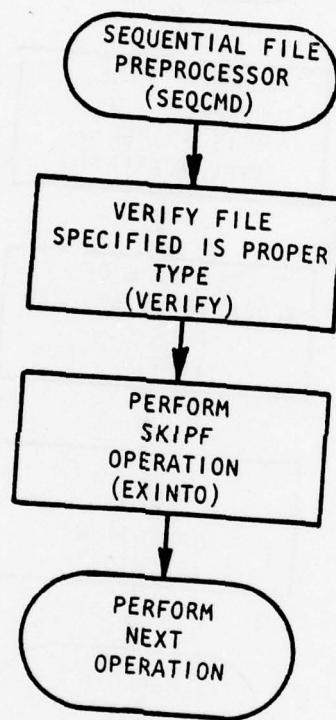


Figure 15. SKIP Function Flowchart

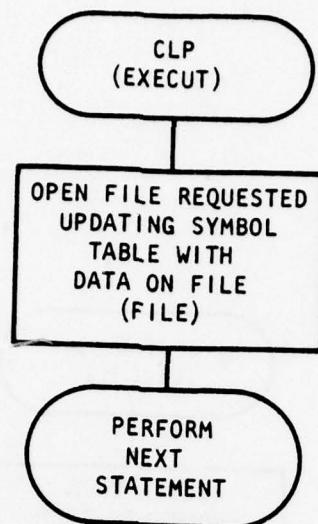


Figure 16. FILE Function Flowchart

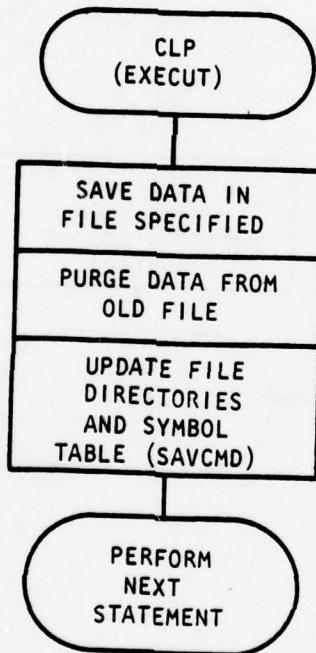


Figure 17. SAVE Function Flowchart

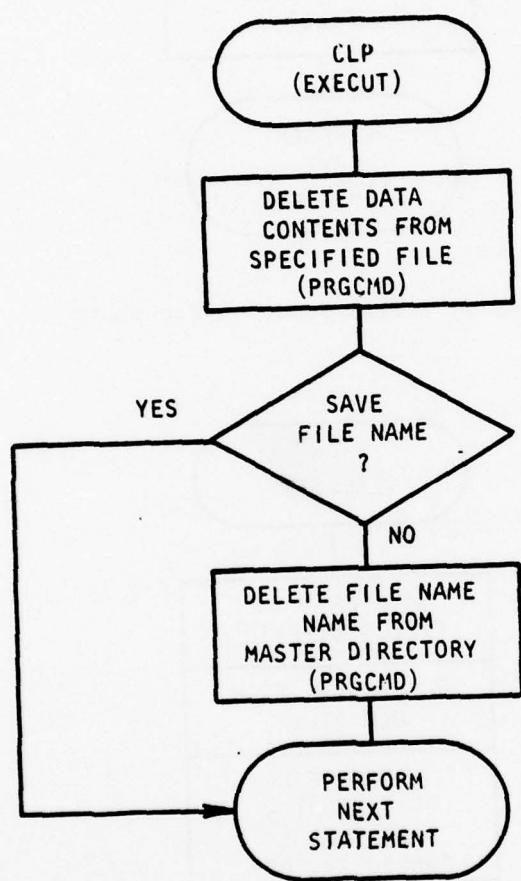


Figure 18. PURGE Function Flowchart

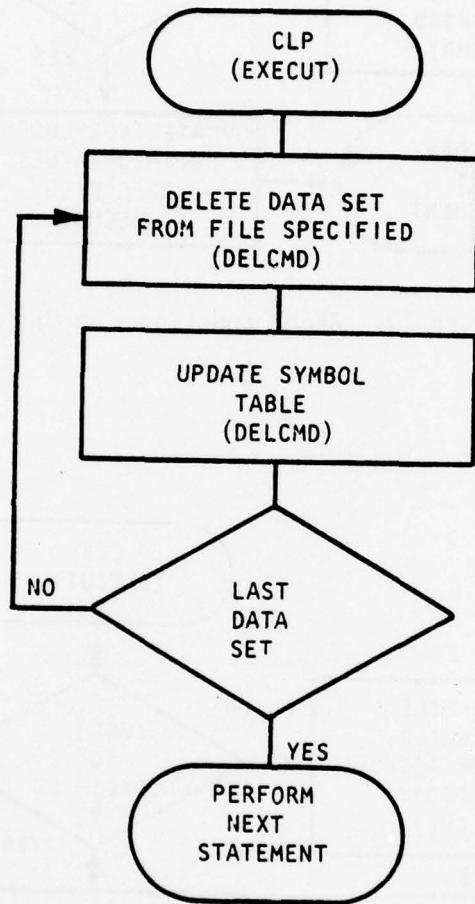


Figure 19. DELETE Function Flowchart

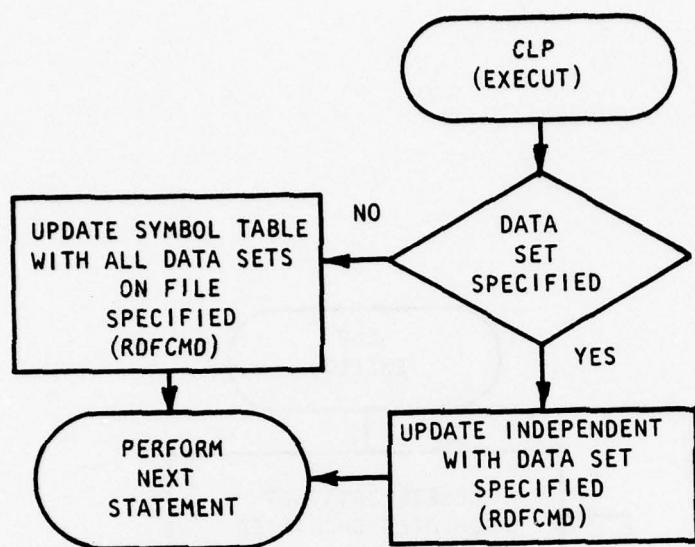


Figure 20. READF Function Flowchart

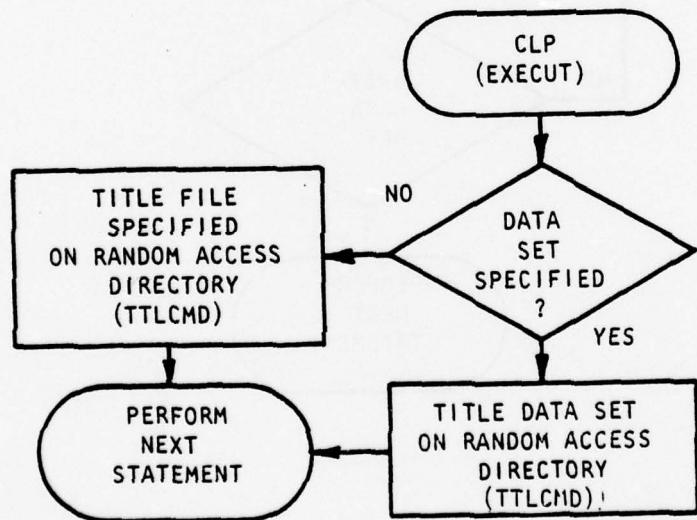


Figure 21. TITLE Function Flowchart

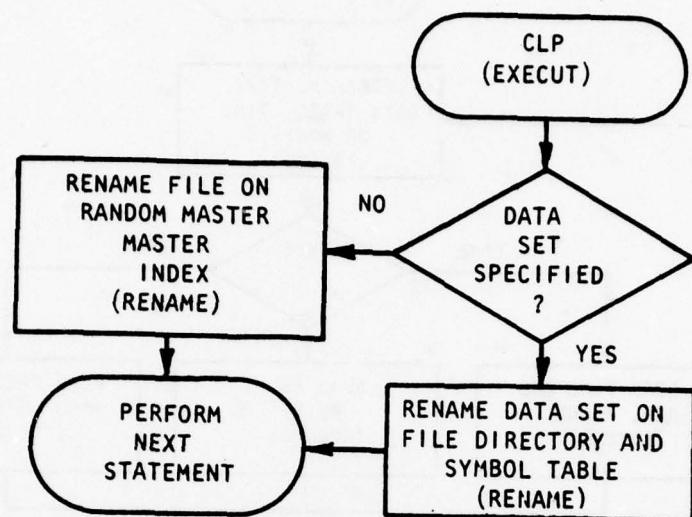


Figure 22. RENAME Function Flowchart

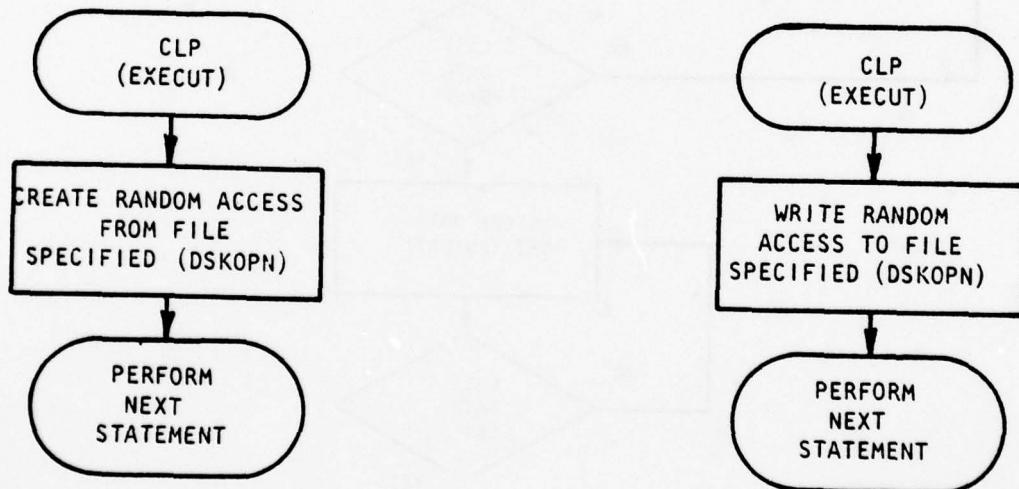


Figure 23. GETRAN and SAVRAN Function Flowcharts

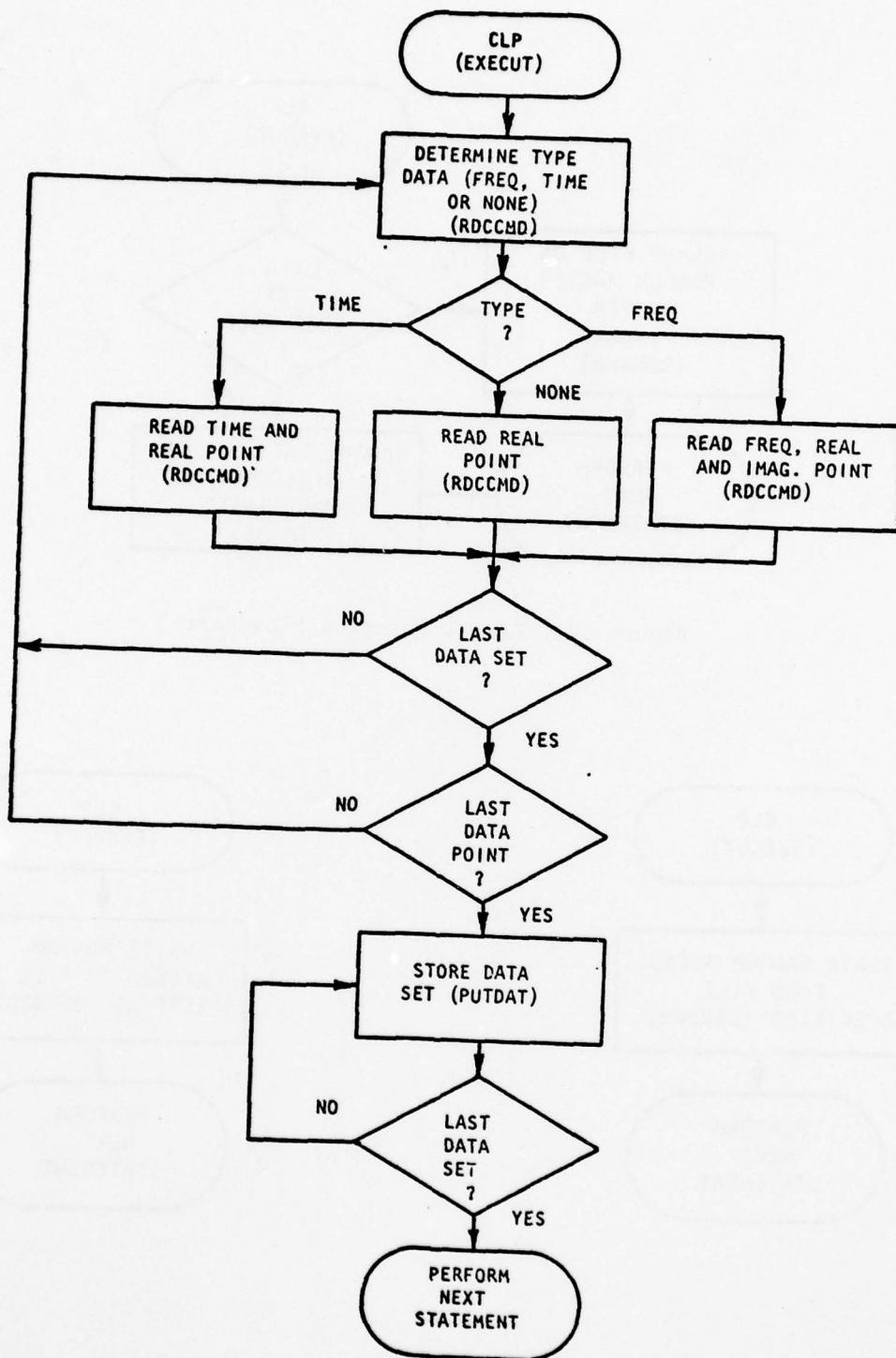


Figure 24. READC Function Flowchart

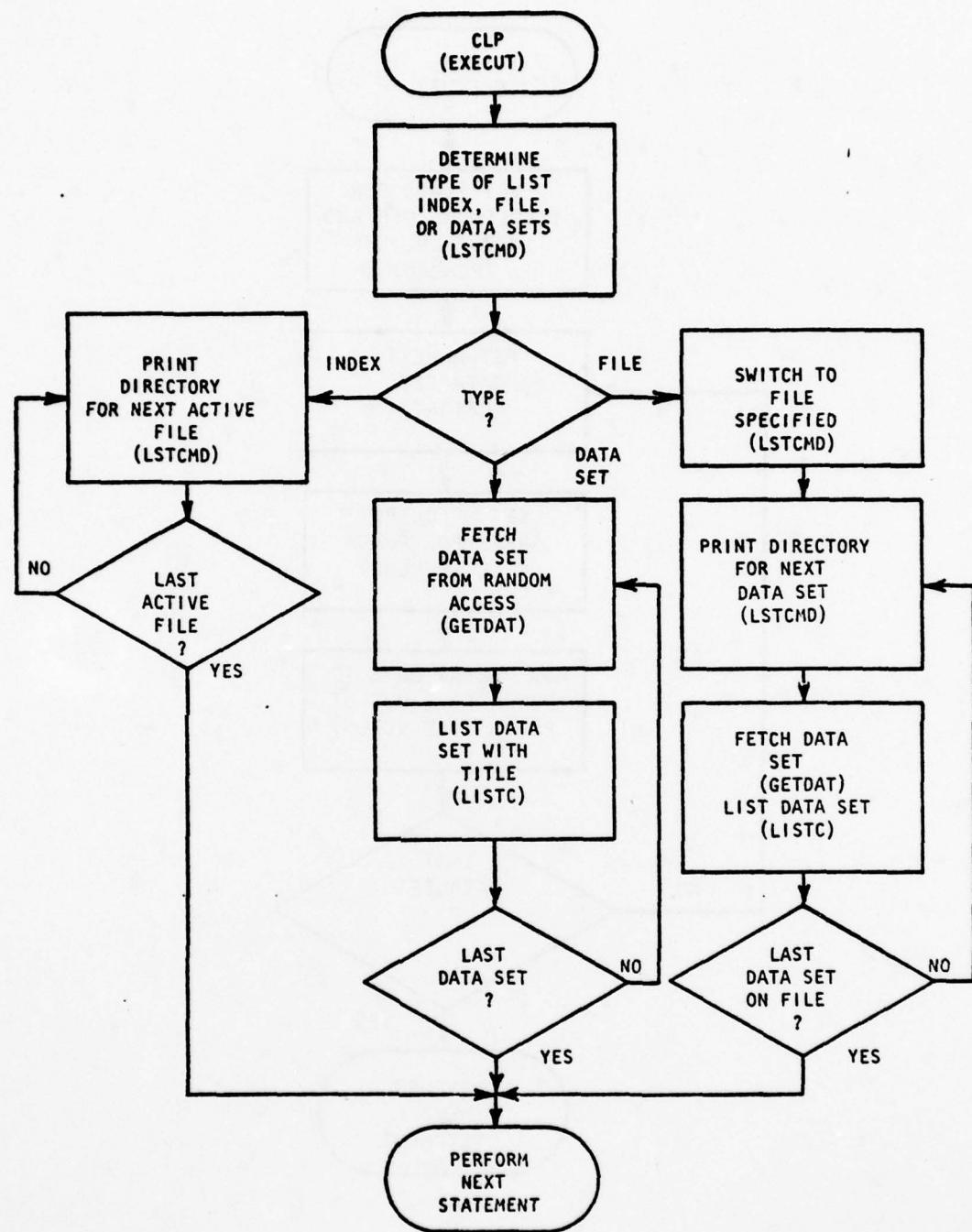


Figure 25. LIST Function Flowchart

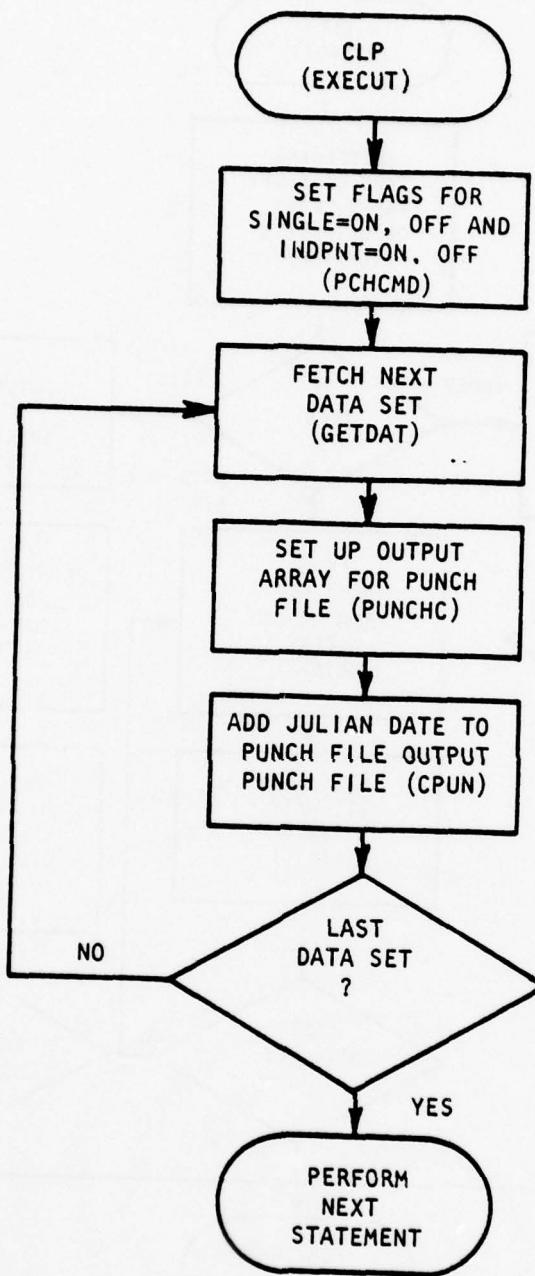


Figure 26. PUNCH Function Flowchart

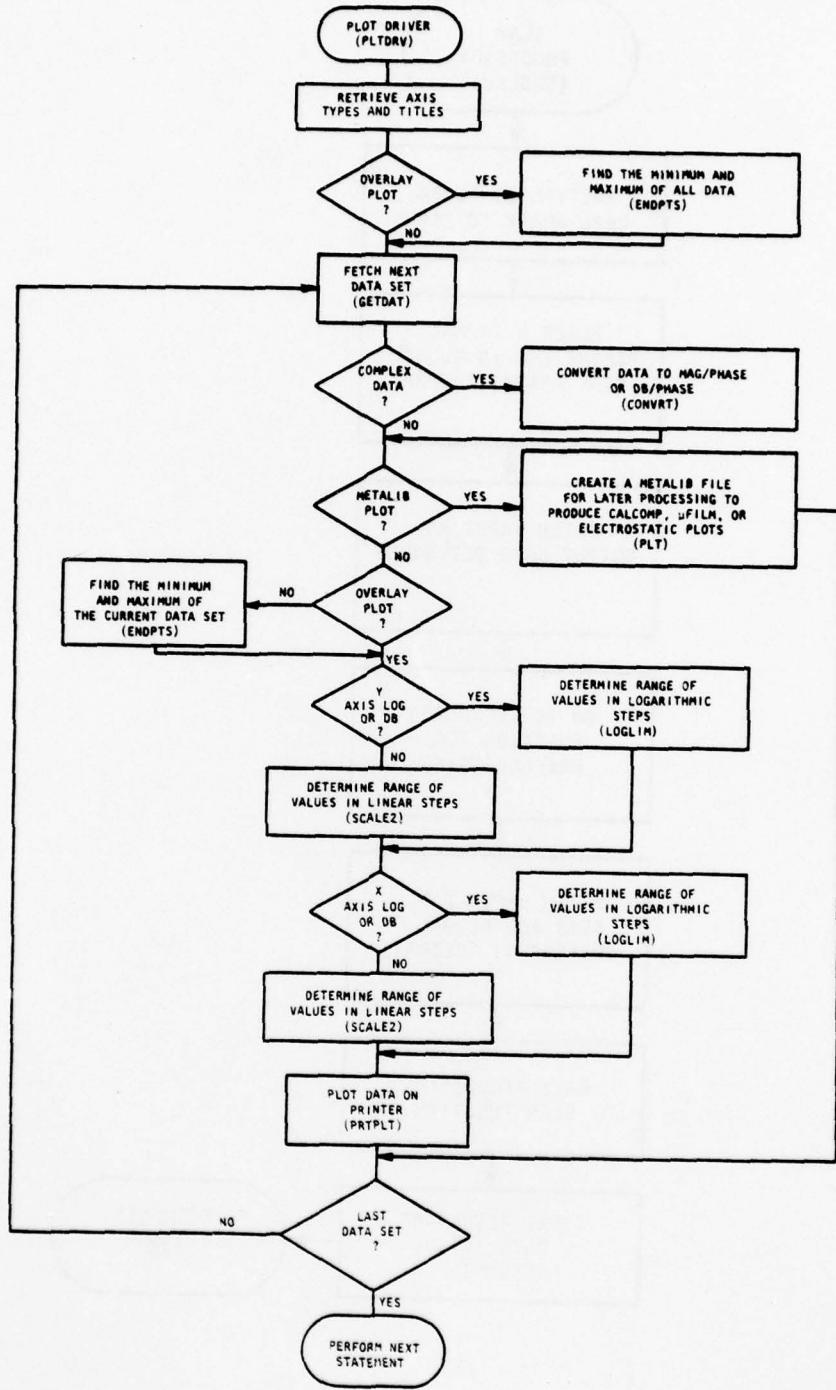


Figure 27. PLOT Function Flowchart

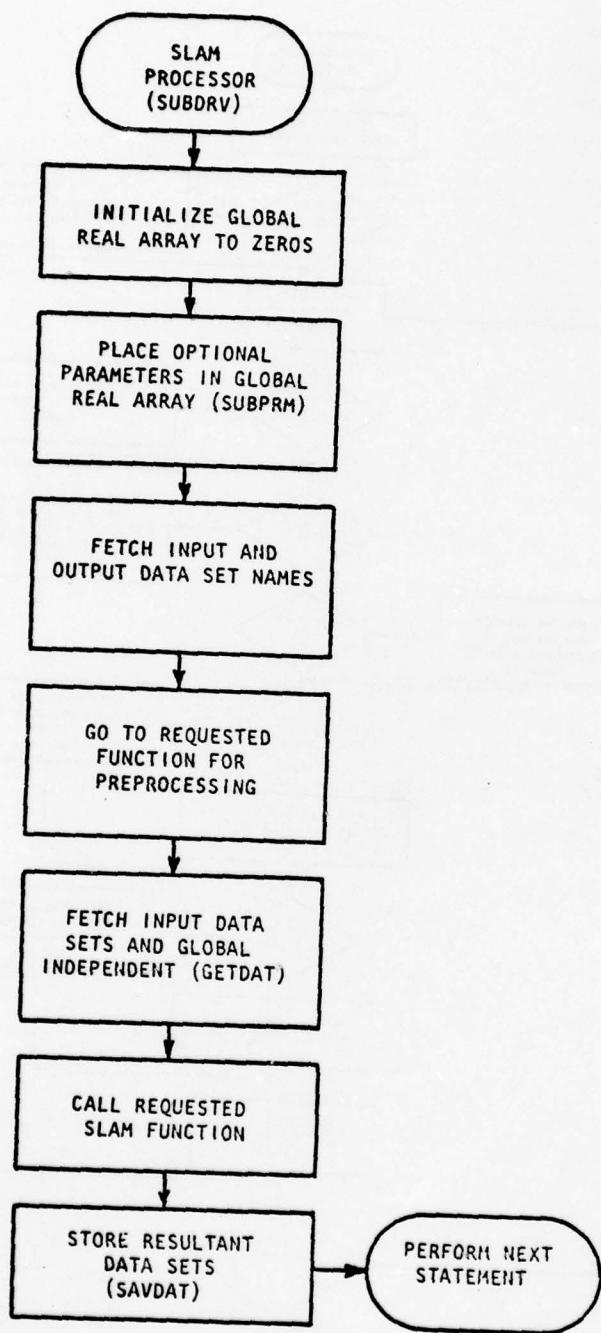


Figure 28. SLAM Function Flowchart

AD-A066 807 BDM CORP ALBUQUERQUE N MEX
EXEMPT PROGRAMMERS MANUAL. (U)
JAN 79

F/G 20/14

UNCLASSIFIED

BDM/A-77-098-TR-R2

AFWL-TR-77-206

F29601-76-C-0122

NL

2 OF 2
AD
A066807



END
DATE
FILMED
5-79
DDC

SLAM function has been completed, the output arrays are stored in the random access system via the subroutine SAVDAT and control returns to the CLP for processing the next input statement.

3. ARITHMETIC, FORTRAN, OR EXEMPT FUNCTIONS

All ECL containing an equal sign as the second entry and those containing the CALL statement are processed in EXPDRV. The function of this processor is illustrated in Figure 29. The argument list is a RPN stack, thus the operator follows the operands. As each operator and its operands are retrieved and executed, their positions in the stack are set to zero. If an operator has a parameter list, these positions are also set to zero. After the operator is determined, the number of parameters and operands are retrieved from the NTSFMT array. The parameter list is loaded on the table immediately preceding the operator and the next operands encountered going back up the stack are associated with the operator. The result of any operator is saved in a temporary array for use with the next operation. If the resultant is not used, it is stored as a symbol with a negative integer for the name. This will not conflict with user specified symbols since they will always be represented as positive integers within EXEMPT.

If the operator corresponds to an EXEMPT library function, any resultant data are stored and control is transferred to the appropriate interface routine to retrieve the data and perform the operation. Upon return to EXPDRV, the independent data will be stored unless a previous operation defined the independent data.

For all other operations, EXPDRV will retrieve the operands and will identify the first independent data set encountered as the independent data to be associated with the expression's resultant data set. Subroutine ARITHOP is called for all FORTRAN and arithmetic operations. When the replacement operator (=) is encountered, the data associated with operands are retrieved and stored as the dependent data of the resultant data set name. If an independent array has been identified, then the resultant data set is properly typed and the independent data retrieved and restored as the independent data of the resultant data set name.

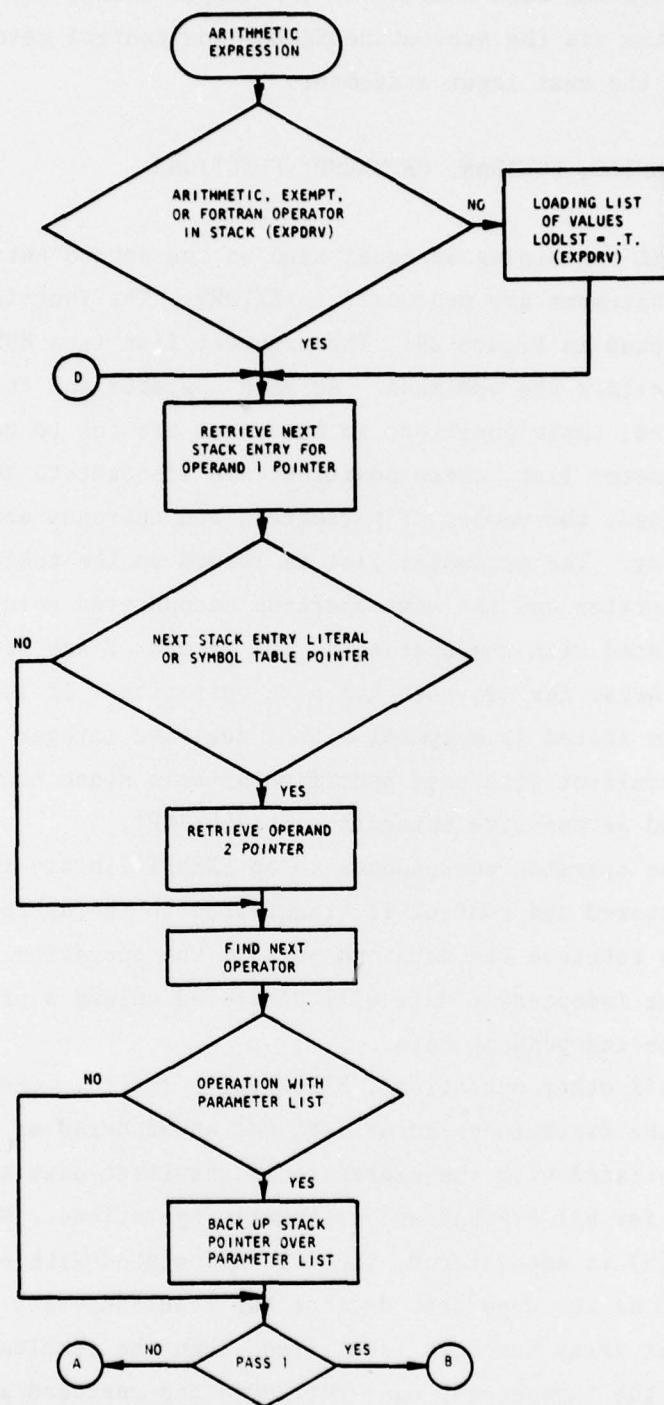


Figure 29. ARITHMETIC Function Flowchart

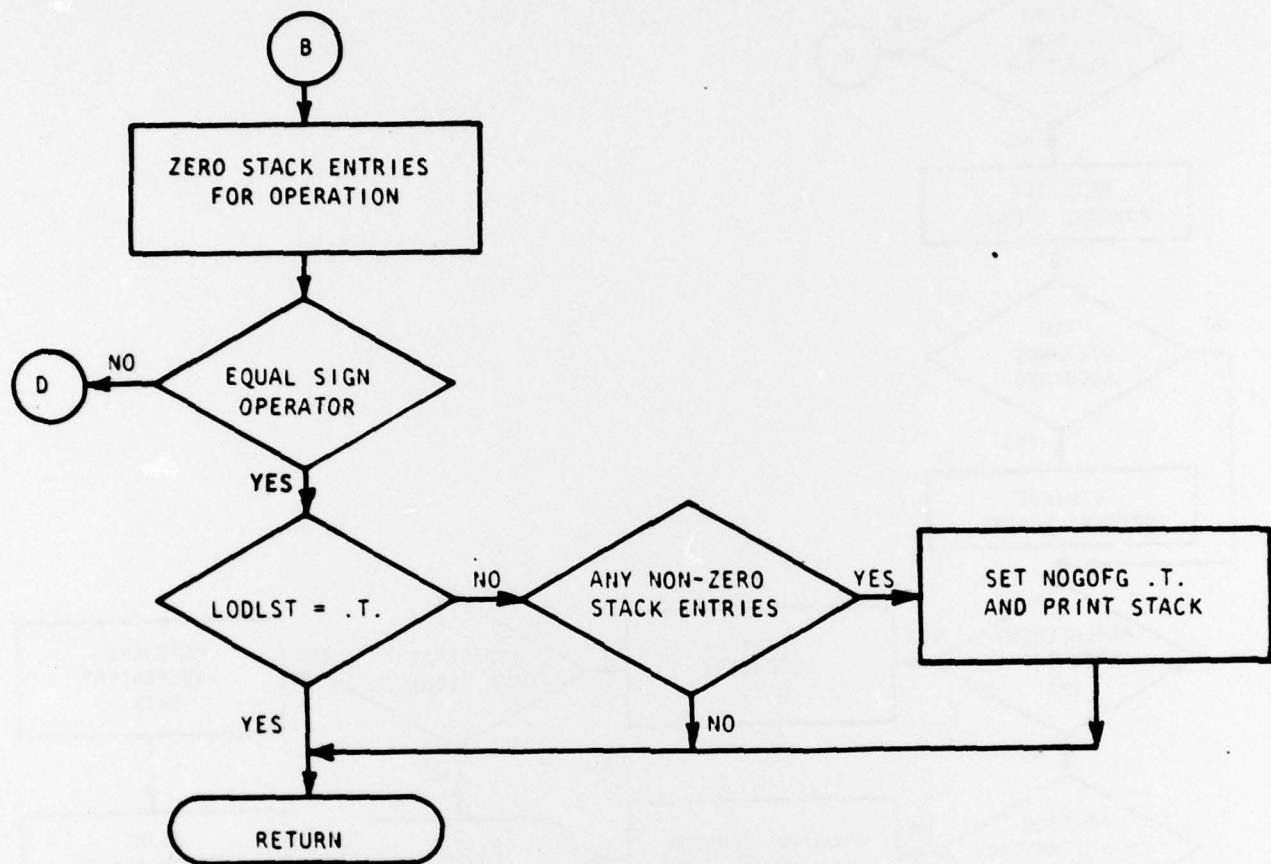


Figure 29. ARITHMETIC Function Flowchart (Continued)

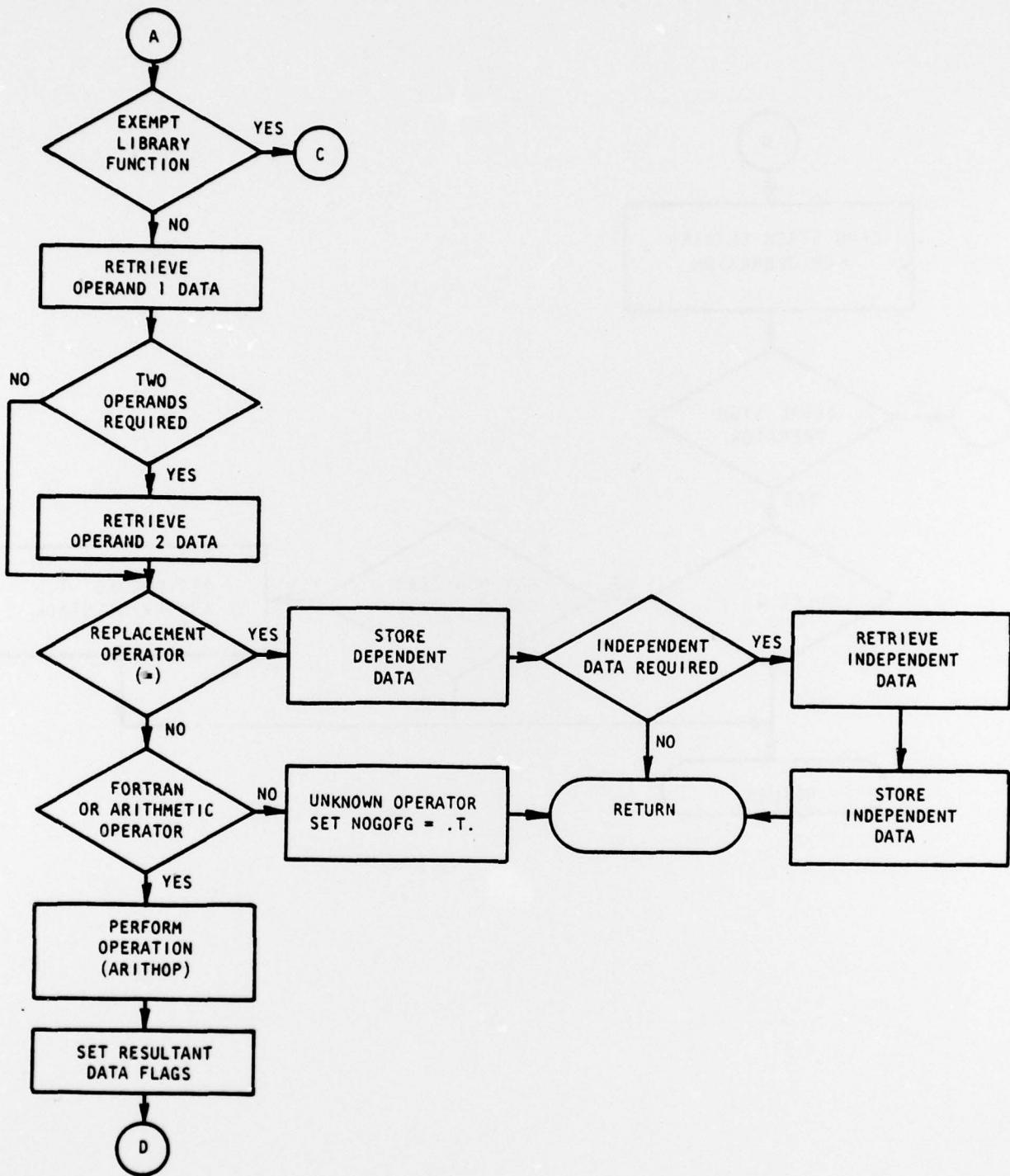


Figure 29. ARITHMETIC Function Flowchart (Continued)

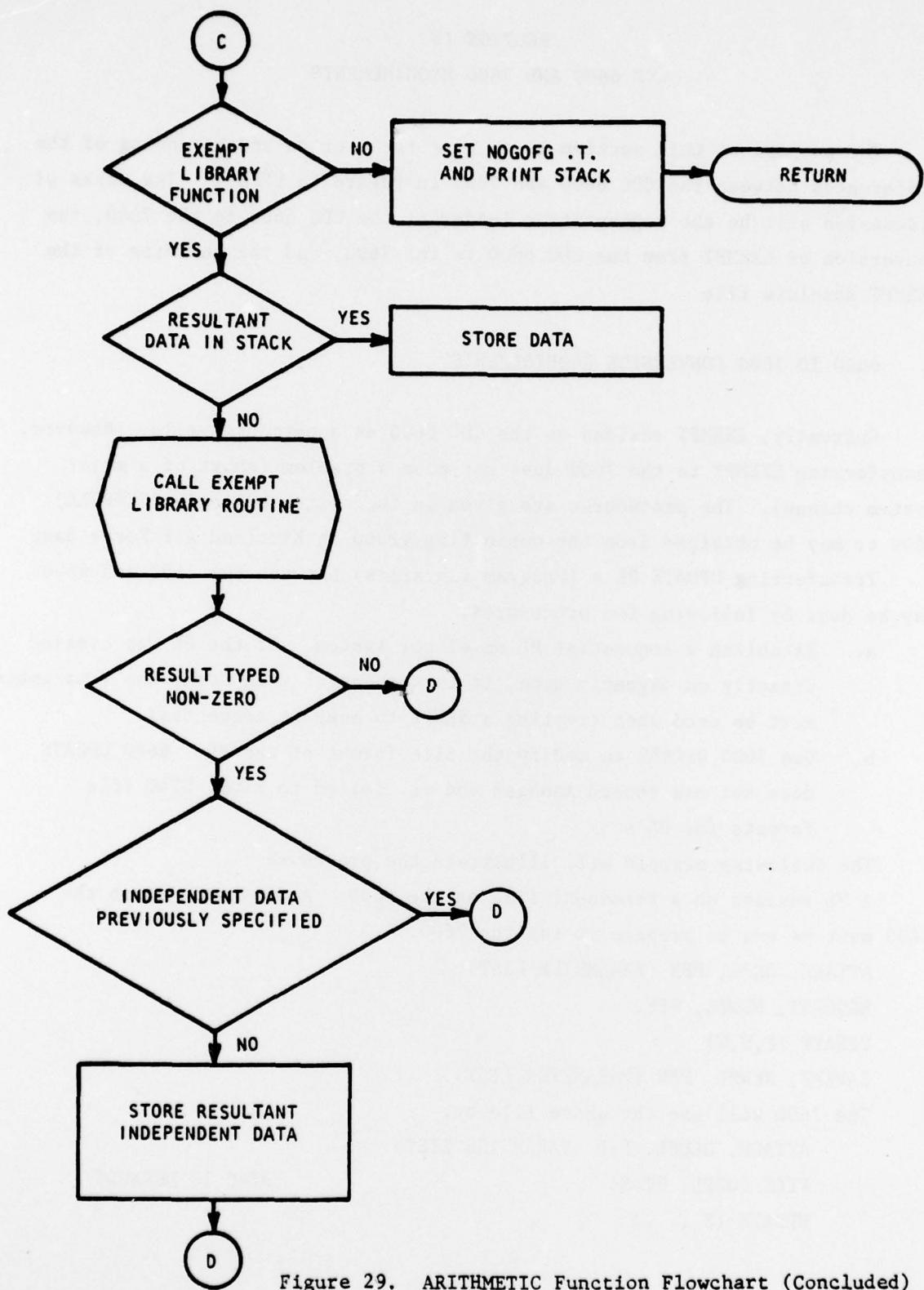


Figure 29. ARITHMETIC Function Flowchart (Concluded)

SECTION IV
CDC 6600 AND 7600 REQUIREMENTS

The purpose of this section is to give the user an understanding of the differences between the CDC 6600 and 7600 in regard to EXEMPT. The areas of discussion will be the segmentation loader of the CDC 6600 to the 7600, the conversion of EXEMPT from the CDC 6600 to the 7600, and the building of the EXEMPT absolute file.

1. 6600 TO 7600 CONVERSION REQUIREMENTS

Currently, EXEMPT resides on the CDC 6600 as a permanent file. However, transferring EXEMPT to the 7600 does not pose a problem (short of a major system change). The procedures are given in the system bulletin (SYSBULL) 7600 or may be obtained from the consulting group at Kirtland Air Force Base.

Transferring UPDATE PL's (Program Libraries) between the 7600 and 6600 may be done by following two procedures:

- a. Establish a sequential PL on either system. If the PL was created directly on magnetic tape, it is sequential. On disk, the W parameter must be used when creating a NEWPL to make it sequential.
- b. Use 7600 UPDATE to modify the file format of the PL. 6600 UPDATE does not use record manager and is limited to RT=S, BT=C file formats for PL's.

The following example will illustrate the procedure.

A PL resides on a permanent file on the 6600. A separate job on the 6600 must be run to prepare it for the 7600.

ATTACH, ODPL, PFN (PARAMETER LIST).

REQUEST, NEWPL, *PF.

UPDATE (F,N,W)

SAVEPF, NEWPL, PFN (PARAMETER LIST).

The 7600 will use the above file by:

ATTACH, OLDPL, PFN (PARAMETER LIST).

FILE (ODPL, RT=S)

BT=C IS DEFAULT

UPDATE (F,....)

2. 6600 AND 7600 SEGMENTATION DIRECTIVES

The primary differences between the two systems is how each loads the system support routines and common blocks. As can be seen by comparing the Figures 30 and 31, the differences are significant. Only two system common blocks need be loaded on the 7600; whereas on the 6600 the main segment must include almost every system routine referenced.

If the user wishes to change to the segmentation directives, the CDC loader manuals should be consulted first for the proper formatting of the directives available.

3. BUILDING EXEMPT ABSOLUTE FILE

Once the user has the OLDPL and the segmentation directives, the EXEMPT absolute file may be created. The first requirement is to make the EXEMPT OLDPL a binary file. This may be done by:

```
ATTACH, OLDPL, PFN (PARAMETER LIST).
UPDATE (F, N)
REQUEST, LGO, *PF.
FTN (I, S=SYSTEXT, S=IPTEXT)
CATALOG, LGO, PFN (PARAMETER LIST).
```

Once the binary has been created then the absolute file may be created as shown in Figure 32. However, if the user wishes to use COPYL SCOPE utility (in the case of modifying or adding a user defined subroutine) then the job stream flow of Figure 33 should be followed. Note that the METALIB library is always loaded. This is due to the fact that the EXEMPT plot routines reference the METALIB routines.

```

*
*
*      MAIN SEGMENT EXEMPT ROUTINES
*
EXEMPT INCLUDE R_KDAT,PLTDAT
*
*      MAIN SEGMENT SYSTEM ROUTINES
*
EXEMPT INCLUDE GET,WA,OPEN,WA,CLSF,WA,PUT,WA,PTWR,SQ,GTWR,SQ
EXEMPT INCLUDE SKHL,SQ,SKSB,SH,CHEK,RM,OPES,SQ,SKSF,SQ
EXEMPT INCLUDE SKFF,SQ,SKFP,SQ,WTMK,SQ,W,SQ,MEM6RM
EXEMPT INCLUDE DT,SQ,R,SQ
EXEMPT INCLUDE PLOTS,GET,RM
*
*      RUN TREE
*
RUN TREE EXEMPT-(CLP,INPDRV)
*      SLAM TREE
*
SLAM TREE SUBDRV-(AS1918,AVRAY,COAX,COCKPT,CONDUIT,ENGINE,HFANT,INDCOU,M
,ARBEA,PITOT,RADALT,RADAR,RVBEA,SHIELD,SNGLIN,WEABAY,WEBAOP,WROOT,WWELL,
,WWGND,SFIELD,AVGSKT,PNLJNT,COUPLER)
*
*      DIRECT MANIPULATION PROCESSOR TREE
*
DMP TREE EXPDRV-(ARITHOP,EDTDRV,FTFDHV,FUNDRV,LODLST,MRGDRV,NTRPDR,OMEGA
,PHZSHF,TRDEXP,TRDSIN)
*
*      PLOT TREE
*
PLTFIL TREE PLTDRV-(OPENPP,PLT,LSERR,PRTPLT,SCALE2)
*
*      COMMAND LANGUAGE PROCESSOR TREE
*
CLP TREE EXECUT-(DMP,SLAM,PLTFIL,EXINTO,PCHCMD,DSKOPN,DSKCLS,RDCCMD,SAVC
,MD,TTLCMD,PRGCMD,RENAME,DELCMD)
*
*
EXEMPT GLOBAL KWNAM1,KWNAM2,KWNAM3,KWNAM4,KWNAM5,FMTPTR,DEFALT
EXEMPT GLOBAL DEFLST,ARGTYP,LOCDAT,TMPFIL,ERR,OUTFL,KWTASK
EXEMPT GLOBAL HOLSTR,ERMODE,ARGTBL,MASK,KEYWRD,MSTRG,DSKNAM,FITS,CHAR
EXEMPT GLOBAL XQTTBL,SYMDAT,PARTBL,SCNTBL,ANGDAT
EXEMPT GLOBAL PLTFMT,PLTLAB,PLTLGV,PLTCM1,PLTCM2,PLTCM3,PLTCM4
*
*      GLOBAL SYSTEM COMMON BLOCKS
*
EXEMPT GLOBAL WR,IO,,FCL,C,,PUT,FO,REW,FD,CLSF,FO,OPEN,FO,GET,FO
EXEMPT GLOBAL CON,RM,AOR,RM
EXEMPT GLOBAL MEMC,RM,IO,BUF.
*
EXEMPT GLOBAL PLTCOM
*
*      END EXEMPT

```

Figure 30. 6600 Segmentation Directives

```

*
*      MAIN SEGMENT EXEMPT ROUTINES
*
EXEMPT INCLUDE BLKDAT,PLTDAT
*
*      RUN TREE
*
RUN TREE EXEMPT-(CLP,INPDRV)
*      SLAM TREE
*
SLAM TREE SUBDRV-(AS1918,AVBAY,COAX,COCKPT,CONDUIT,ENGINE,HFANT,INDCOU,M
*      ARBEA,PITOT,RADALT,RADAR,RVBEA,SHIELD,SNGLIN,WEABAY,WEBAOP,WROOT,WWELL,
*      WWGND,SFIELD,AVGSKT,PNLJNT,COUPLER)
*
*      DIRECT MANIPULATION PROCESSOR TREE
*
DMP TREE EXPDRV-(ARITHOP,EDTDRV,FTFDRV,FUNDRV,LOOLST,MRGDRV,NTRPDR,OMEGA
*      ,PHZSHF,TRDEXP,TRDSIN)
*
*      PLOT TREE
*
PLTFIL TREE PLTDRV-(OPENPP,PLT,LSERR,PRTPLT,SCALE2)
*
*      COMMAND LANGUAGE PROCESSOR TREE
*
CLP TREE EXECUT-(DMP,SLAM,PLTFIL,EXINTO,PCHCMD,DSKOPN,DSKCLS,RDCCMD,SAVC
*      ,MD,TTLCMD,PRGCMO,RENAME,DELCMD)
*
*
EXEMPT GLOBAL KWNAM1,KWNAM2,KWNAM3,KWNAM4,KWNAM5,FMTPTR,DEFALT
EXEMPT GLOBAL DEFLST,ARGTYP,LOCDAT,TMPFIL,ERR,OUTFL,KWTASK
EXEMPT GLOBAL HOLSTR,ERMODE,ARGTBL,MASK,KEYWRD,MSTRG,DSKNAM,FITS,CHAR
EXEMPT GLOBAL XQTBL,SYMDAT,PARTBL,SCNTBL,ANGDAT
EXEMPT GLOBAL PLTFMT,PLTLAB,PLTLCV,PLTCM1,PLTCM2,PLTCM3,PLTCM4
*
*      GLOBAL SYSTEM COMMON BLOCKS
*
EXEMPT GLOBAL FCL,C,QB,IO.
*
EXEMPT GLOBAL PLTCOM
*
*      END EXEMPT
*

```

Figure 31. 7600 Segmentation Directives

JOB CARD.
ACCOUNT CARD.
ATTACH•OLDPL•PFN (PARAMETER LIST).
FILE(OLDPL,RT=S)
UPDATE(F,N)
RETURN•OLDPL.
REQUEST•LGO•*PF.
FTN(I,S=SYSTEXT,S=IPTEXT)
CATALOG•LGO•PFN (PARAMETER LIST).
RETURN•LGO.
REQUEST•EXMABS•*PF.
ATTACH•EXMSEG•PFN (PARAMETER LIST).
ATTACH•LGO•PFN (PARAMETER LIST).
LIBRARY(METALIB)
SEGLOAD•I=EXMSEG•B=EXMABS.
LOAD(LGO)
NOGO.
CATALOG•EXMABS•PFN (PARAMETER LIST).
S (EOR) UPDATE DIRECTIVES
(EOF) END OF JOB

Figure 32. Creating an EXEMPT Absolute File

JOB CARD.
ACCOUNT CARD.
FTN.
ATTACH,OLD,PFNAME (PARAMETER LIST). EXEMPT BINARY
COPYL,OLD,LGO,EXMBIN,,A.
LIBRARY(METALIB)
ATTACH,SEG,PFNAME (PARAMETER LIST). SEGMENT DIRECTIVES
SEGLOAD(I=SEG)
LOAD(EXMBIN)
NOGO.
ABS.
\$ 7/8/9 (EOR) INPUT FOR FTN CARD
\$ 7/8/9 (EOR) EXEMPT ECL
6/7/8/9 (EOF) END OF JOB

Figure 33. Absolute File With a User-Defined Subroutine